

A TRIDENT SCHOLAR PROJECT REPORT

NO. 238

"DUAL PURKINJE-IMAGE EYETRACKER"



UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

This document has been approved for public
release and sale; its distribution is unlimited.

DTIC QUALITY INSPECTED 1

20000406 106

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget Paperwork Reduction Project (0704-0188), Washington, DC 20503.

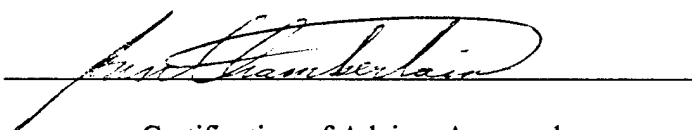
1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE 1996	3. REPORT TYPE AND DATES COVERED	
4. TITLE AND SUBTITLE Dual Purkinje-image eyetracker			5. FUNDING NUMBERS	
6. AUTHOR(S) Chamberlain, Ann C.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) United States Naval Academy Annapolis, MD 21402			10. SPONSORING/MONITORING AGENCY REPORT NUMBER USNA Trident Scholar report; no. 238 (1996)	
11. SUPPLEMENTARY NOTES Accepted by the U.S.N.A. Trident Scholar Committee				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE UL	
13. ABSTRACT (Maximum 200 words) This project presents a new tool to track the eye's angle of rotation. The method employed is that of tracking two reflections, the first and fourth Purkinje-images. Four Purkinje-images are formed within the eye as light is reflected off of corneal and eye lens surfaces. Coincidentally, the first and fourth reflections occur on the same image plane allowing them to be simultaneously captured by a focused charge-coupled device (CCD) camera. Measurements of the relative displacement between these two images reveal the orientation of the eye in space independent of head position in space. An infrared point light source was used to create the Purkinje-images within the eye and a CCD camera was used to capture the images. Computer hardware and software were used to record the eye images and to analyze the image data. Image processing techniques were successfully employed to locate two Purkinje-images within each image of the eye. Through the use of image processing techniques, it is shown that it is feasible to develop a more efficient and lightweight system that is accurate, non-invasive to the subject, and allows the subject to be mobile. Eye tracking systems find application in aids to disabled patients, medical procedures, and military systems. Adapting this work to real-time Eye tracking is a future consideration.				
14. SUBJECT TERMS Eye tracker, Eye movements, Dual Purkinje-image, Image processing			15. NUMBER OF PAGES 65	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

U.S.N.A. --- Trident Scholar project report, no. 238 (1996)

"DUAL PURKINJE-IMAGE EYETRACKER"

by

Midshipman Ann C. Chamberlain, Class of 1996
United States Naval Academy
Annapolis, Maryland



Certification of Adviser Approval

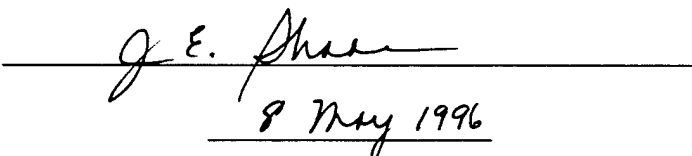
Assistant Professor Carl E. Wick
Department of Weapons & Systems Engineering



5 May 1996

Acceptance for the Trident Scholar Committee

Professor Joyce E. Shade
Chair, Trident Scholar Committee



8 May 1996

USNA-1531-2

ABSTRACT

This project presents a new tool to track the eye's angle of rotation. The method employed is that of tracking two reflections, the first and fourth Purkinje-images. Four Purkinje-images are formed within the eye as light is reflected off of corneal and eye lens surfaces. Coincidentally, the first and fourth reflections occur on the same image plane allowing them to be simultaneously captured by a focused charge-coupled device (CCD) camera. Measurements of the relative displacement between these two images reveal the orientation of the eye in space independent of head position in space. An infrared point light source was used to create the Purkinje-images within the eye and a CCD camera was used to capture the images. Computer hardware and software were used to record the eye images and to analyze the image data. Image processing techniques were successfully employed to locate two Purkinje-images within each image of the eye.

Through the use of image processing techniques, it is shown that it is feasible to develop a more efficient and lightweight system that is accurate, non-invasive to the subject, and allows the subject to be mobile. Eyetracking systems find application in aids to disabled patients, medical procedures and military systems. Adapting this work to real-time eyetracking is a future consideration.

Keywords: Eyetracker, Eye Movements, Dual Purkinje-Image, Image Processing

ACKNOWLEDGMENTS

I would like to express thanks to my advisor Professor Carl E. Wick whose patience and instruction guided me through this program. Thanks also to my parents Robert and Iracema Chamberlain. Thanks to Carl Owen in the Machine Shop and to Major Bonsignore of the Computer Science Department. And finally, thanks to my friends who helped in seeing me through to the end of this project.

TABLE OF CONTENTS

Abstract	1
Acknowledgments	2
Table of Contents	3
List of Illustrations	4
Section 1: Background	5
Section 2: Eyetracking Systems	10
Section 3: Existing Dual Purkinje Systems	22
Section 4: Experimental Apparatus	25
Section 5: Image Processing Algorithms	31
Section 6: System Performance	41
Section 7: Conclusion	43
Section 8: Future Activities	44
References	45
Bibliography	46
Appendix-Program Source Code	48

LIST OF ILLUSTRATIONS

Figure 1:	Electro-Oculography	10
Figure 2:	Eyetracking Method Used by the EWP	11
Figure 3:	Apparent Pupil Ellipticity	12
Figure 4:	Limbus Tracking Scan	13
Figure 5:	Clam Shell Structure of the Optical Surfaces of the Eye	16
Figure 6:	Four Purkinje Images	18
Figure 7:	Photograph of the Generation-V Dual Purkinje Image Eyetracker	23
Figure 8:	Schematic of the Generation-V Eyetracking System	23
Figure 9:	Photograph of the System Designed in this Project	25
Figure 10:	Block Diagram of the Project System	26
Figure 11:	Schematic of the Helmet-Mounted System	29
Figure 12:	Photograph of the Helmet-Mounted System	29
Figure 13:	Photograph of Eye Glints	32
Figure 14:	Histogram of Eye Image	34
Figure 15:	Successfully Binarized Eye Images	35
Figure 16:	Results of a Majority Filter	36
Figure 17:	Two Successfully Processed Images	39

SECTION 1: Background

The potential applications of an eyetracker are so numerous that it is not practical to list them all in a single paper. Yet, although they are many, they may not be readily called to mind. Therefore, to familiarize the reader with the purpose for developing an accurate and effective eyetracker, a few practical applications for an eyetracker follow.

An immediate application exists in the forefront of modern medicine. Tracking eye movements in real time is a prerequisite for computer-assisted laser positioning in diagnosis or in therapy of the eye. The development of an automatic, real-time eyetracker would enable a surgeon to keep a laser beam correctly positioned on target while performing corneal surgery despite fine adjustments made by the eye, which is continuously moving, even when gazing. Correcting for these unpredictable micro eye movements would allow corrective procedures in eye surgery to become more accurate, enabling surgery to be conducted closer to the pupil and with less risk of damage to the eye. If surgery is performed too close to the center of vision a sudden eye movement could bring the laser beam onto the fovea of the eye and cause serious visual damage. Currently the risk involved in eye surgery is great. With an eyetracker capable of accurately controlling a laser in real-time, surgery to correct myopia, hyperopia and astigmatism could all become safer endeavors. The sudden movements of the eye would be automatically compensated for by the eyetracker [1].

A second application deserving mention is the use of an eyetracker as an aid to victims affected by the incurable disease, amyotrophic lateral sclerosis (ALS). Patients with ALS suffer by gradually losing the ability to move all muscles until only eye

movements remain. Development of an effective, accurate, and rapid eyetracker would improve the ability and extent with which these patients can communicate and interact with those around them. An eyetracker working in conjunction with an "Eye Word Processor," such as the one developed by Yamada [2], can furnish a victim of ALS with the means to write sentences and control peripheral equipment like electric lights, a television and a telephone call system. Yamada's system works by presenting the ALS victim with a screen of letters on a monitor. A calibrated eyetracker then provides a processor with information about the location of eye gaze. The area on the screen within the user's gaze is assumed to contain the user's input letter or request. The user is then free to input subsequent letters, etc. [2].

Another function of an eyetracker is to conduct investigations in order to achieve a greater understanding of eye mechanics. One investigation undertaken in this pursuit was in an attempt to explain retinal image distortions that occur during eye pursuit motions. Another similar investigation was conducted to determine the differences in pursuit operations of the eye between human subjects in two groups; one group with intact interaction across the two hemispheres of the brain, and one group who had this connection surgically severed. The eye motions in both investigations were tracked using a Purkinje image eyetracker. Specialized software was developed to analyze the characteristics of pursuit eye motions. In this application, an eyetracker facilitated investigations of basic visual and oculomotor processes [3].

It is also possible to study the essential mechanical abilities of the eye by examining specific types of eye motions. For example, optokinetic nystagmus is an eye

motion that can be described as the movements involved when the eye fixates on a portion of a moving field and follows it for a period of time with pursuit motion, then jumps (saccades) to fixate on a new portion of the field. Attempts have been made to use optokinetic nystagmus as an objective measurement of visual acuity by determining the minimum line width that induces nystagmus. These investigations were made in order to explore and determine the mechanical limitations of the human eye [4].

The study of specific eye motions can also be used in examining the function of various physiological systems. Vestibular nystagmus is an oscillatory motion of the eye much like optokinetic nystagmus, containing a slow phase and a fast saccadic-like return. This specific eye motion is attributable to stimulation of the semicircular canals during rotation of the head with respect to inertial space. Measurement of vestibular nystagmus in various axes is commonly used to test semicircular canal function, either through the threshold of angular acceleration impulse required to induce nystagmus or through the duration of post-rotation nystagmus. In addition, the measurement of any occurrence of spontaneous nystagmus is often associated with a number of neurological disorders. Thus, disfunction in eye motion can be directly related to mental disfunction. Abnormal nystagmus can also be detected through the use of an eyetracker [4].

Through tracking points of eye gaze within a scene, it is possible to ascertain what visual information within a scene is instinctively deemed important by an individual. The US Army has been interested in discovering how an eye distinguishes a target hidden within a random background of noise. The Army has been conducting studies to determine what gazing points are selected by an individual and which of these points are

given the most importance, as determined by length of gaze. An eyetracker was used in collecting this data. The main objective of the Army investigations was to eventually develop a model eye system equipped with mechanical sensors. The system is to be capable of automatically searching for the same type of target while scanning a similar background. It is hoped that this system would be able to locate and identify a target with an accuracy, efficiency, and speed that could match human eyes in image processing abilities. Through observation and analysis of human visual behavior, a greater understanding of the workings of the human eye develops [5].

An eyetracker could also be used in the development of more accurate weapons targeting systems in the military. The Air Force is currently developing technology for a helmet-mounted eyetracker, which is to be integrated with a helmet-mounted virtual panoramic display (VPD). In this system, eyetracking will be used for eye controlled switch selection, cueing, target designation, and pilot state monitoring. The use of an eyetracker to accomplish tasks in a cockpit improves the pilot's ability to manipulate switches under high g conditions or when the pilots' hands are busy with other tasks. In cueing, a virtual display of the pilot's line of gaze is presented to the copilot and vice versa. This affords each member in a plane to quickly direct his/her gaze to where the other member is looking. There is no need for verbal explanation. For example, once the copilot sights a target, the pilot, aware of the direction the copilot is looking in, will subsequently almost immediately sight the same target. Using an eye tracker to aid in the task of target designation offers the advantages of faster target acquisition and the capability to function in high g environments more accurately than with the use of head

motion tracking. Feedback in the form of a superimposed cross hair on a virtual display will indicate when the weapons system is locked onto the target. The pilot then confirms and launches a weapon with a consent switch or by voice command. In pilot-state monitoring, pupil diameter is tracked as an indication of pilot awareness. However, the reliability in tracking pupil diameter to reflect the pilots's state of alertness, fatigue, stress, and work load has not yet been established [6].

Although the utility of an eyetracker is unquestionable, the development of an effective and accurate system capable of functioning in each and all of the above capacities has been a difficult endeavor. Section 2 follows with a description of several different approaches that have been taken in the research and design process of developing an ideal eyetracker.

SECTION 2: Eyetracking Systems

The process of measuring eye movement has produced a variety of eyetrackers. Each tracker variety performs its task by recording different physical properties related to eye angle of rotation. Some of these properties include an electrical potential difference that exists between the cornea and the retina, an electrical impedance measured

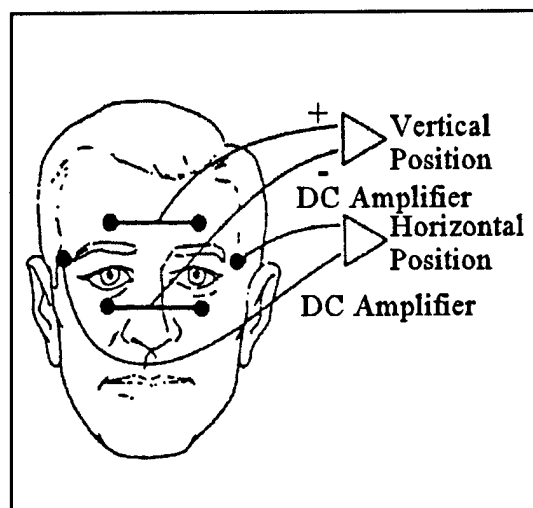


Figure 1. Electro-Oculography

This is a diagram of electrode placement to measure vertical and horizontal eye movement in the technique of Electro-Oculography. Electro-Oculography tracks changing corneoretinal potential [4].

between two separate points on the eye, the position of scleral and retinal blood vessels, the position of the limbus (boundary between the iris and the sclera), the location of the center of the pupil, the location of the corneal bulge, and the position of corneal reflections or other occurring reflections. Several different techniques for tracking eye movement result. The physical properties, the techniques, and the major advantages and disadvantages of many current eyetracking systems are briefly presented in this section of the paper. Two comprehensive sources exist that summarize the various techniques. These sources were referenced frequently throughout this section [4,7].

One eyetracking technique, electro-oculography, is a method that involves the placing of skin electrodes around the eye and recording potential differences in electrical energy resulting from the potential difference that exists between the cornea and the retina. A diagram of electrode placement can be seen in Figure 1. The corneoretinal

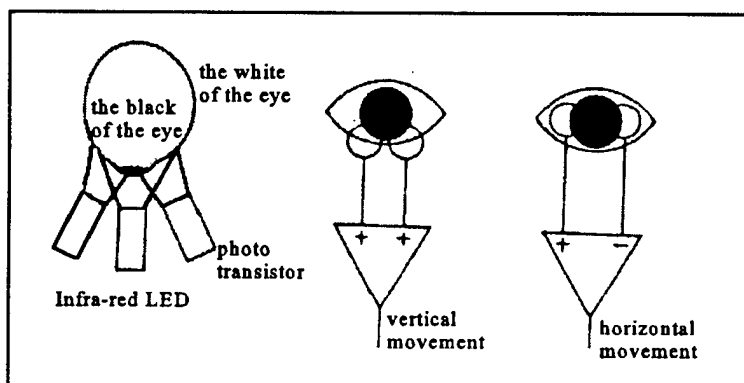


Figure 2. Eyetracking Method Used by the EWP

A method of detecting eye movement through the use of infra-red LEDs and photo detectors. A horizontal movement of the eye results in a differential sensor output. A vertical movement of the eye results in a concurrent sensor output [2].

potential can be described as a vector that rotates with respect to the eye. This method does not require visualization of the eye and is usable for eye movements up to ± 70 degrees. Typical accuracy

with surface electrodes is ± 1.5 -2 degrees, provided eye excursions remain less than ± 30 degrees. A major disadvantage of electro-oculography is an associated dc drift problem which can result in minor expected changes in the dc signal to be mistaken for movements of the eye. Another problem is associated with the fact that the electric field of this potential is not aligned with the optic axis. Therefore, a torsional rotation of the eye may introduce a potential change that might be mistaken for horizontal or vertical eye movement. In addition, the basis of this method is imprecise since the potential has been shown to vary diurnally and with light adaptation [4,7].

It is also possible to measure eye position by measuring the electrical impedance between electrodes placed at the outer temple points of the two eyes. The resistance, either associated with the nonhomogeneous electrical characteristics of the tissues in the eye or with the nonspherical characteristics of the eye, changes between the two electrodes and can be correlated to changes in eye position [4].

The system used to detect eye movements by Yamada [2] in conjunction with the

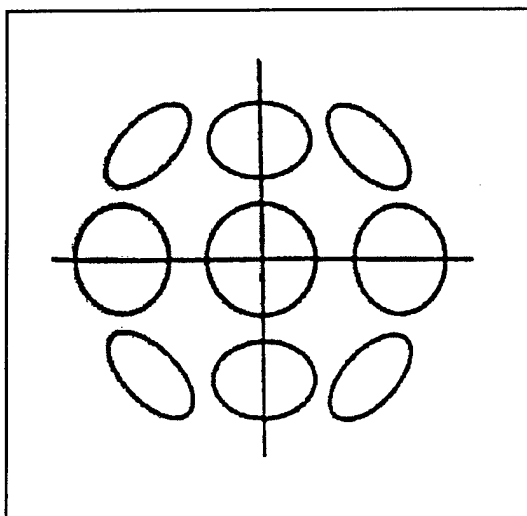


Figure 3. Apparent Pupil Ellipticity
The apparent pupil shape as the eye rotates is an even function. Therefore there is no difference between up and down or right and left movement [4].

Eye Word Processor (EWP), was composed of photo resistors and respective infra-red LEDs placed in a row opposite the eye. Horizontal movements were detected differentially and vertical movements were detected concurrently. Figure 2 illustrates the infra-red LED and sensor set-up with respect to the eye.

One method of eyetracking employs

the fact that the pupil of the eye takes on

various elliptical shapes when viewed at a perspective other than head on, when the pupil appears circular. Figure 3 illustrates the apparent ellipticity of the pupil as the eye rotates. By determining the ellipticity of an eye from an observer's point of view, the rotation angle of the eye may be determined. The major difficulty in the implementation of this method is that ellipticity of the eye as a function of eye rotation angle, is an even function; thus directional ambiguities occur. Some other eye characteristics would have to be used to differentiate between up and down and right and left [4].

Optical landmarks can be measured in tracking eye movement. These include scleral and retinal blood vessels. In addition, artificial landmarks can be placed on the eye and consequently tracked, such as a small piece of metal imbedded in the sclera to be used for magnetic tracking [4].

The location of the iris-scleral boundary (the limbus) can be recorded in order to

measure horizontal position of the eye. The ratio between the dark iris and bright sclera is observed on the left and right sides of the eye, indicating limbus location. The best wavelength for making the distinction between iris and sclera depends to some extent

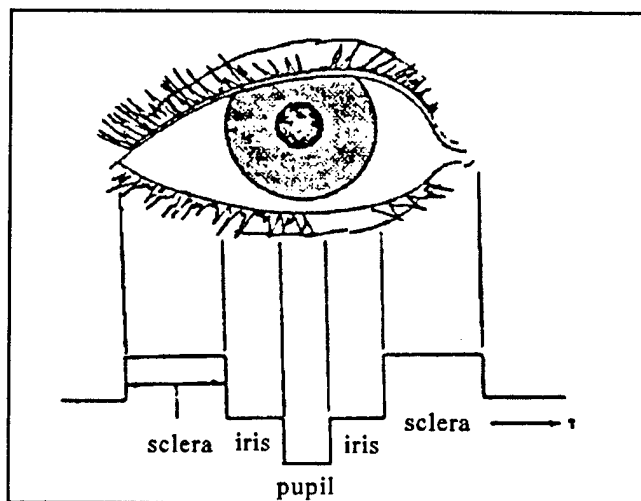


Figure 4. Limbus Tracking Scan

This schematic illustrates one method of scanning an image of the eye to determine the location of the limbus (the iris-sclera boundary) [4].

upon iris color. An ideal horizontal scan of an eye image, measuring eye position through limbus tracking is illustrated in Figure 4. Most of the difficulty in limbus tracking stems from the complex problem of determining vertical eye movement with the existence of an intruding eyelid. If pupil tracking is used in conjunction with limbus tracking, vertical movement can be determined, which then increases the accuracy of subsequent horizontal eye movement measurement. A characteristic of pupil tracking that adds to its complexity is that, due to both psychological and physiological influences, the pupil diameter and circumference are variable. For this reason, pupil circumference or diameter tracking is not used unless pupil size data are required by the experimenter in addition to eye movement data. For head-fixed photo detectors and illuminators, free head movement is possible in limbus tracking and the measurement is relative to head position. Limbus tracking is a single-feature optical detection technique. Therefore, relatively large errors are associated with any movement of the head-fixed

apparatus with respect to the head. Helmet slip error associated with the limbus tracking technique can be over 4 degrees for 1 mm of optics system translation with respect to the head [4,7].

Measurement of pupil location also serves as a basis for eye angle measurement. The pupil can be separated from the surrounding iris optically. This can be more easily accomplished with the use of infrared light, which will be almost entirely absorbed by the pupil, consequently making the pupil much darker than the surrounding iris. Varying between 2 and 8 mm in diameter in adult humans and slightly elliptical in shape, the pupil can be approximated closely by tracing the best-fitting circle to the pupil circumference [4].

Several methods for measuring eye movement by recording the location of the corneal bulge have been developed. The earliest of which involved making attachments to the cornea that were mechanically linked to recording pens. Another method involved placing pressure transducers over the eyelid in order to determine the location of the bulge as it was being felt underneath a closed eyelid. More recently, the cornea has acted as a mechanical post to center tight fitting scleral contact lenses. The contact lens method is considered to be the most precise eye movement measurement technique. Conventional corneal lenses are too mobile, therefore a special contact lens is used. A tight fit and lack of slip in this lens is achieved through negative (suction) pressure. Negative pressure contact lens systems cause discomfort and the very tight ones usually require the application of a topical anesthetic. Some lenses have been constructed to include a protruding stalk. These systems preclude normal blinking responses. The most

commonly used contact lens system is the "optical lever," in which one or more plane mirror surfaces ground on the lens reflect light from a light source to a photo detector. These mirror systems suffer from changes in mirror properties with tear film. Therefore, several contact lens optical mirror systems employ mirrors or lights mounted on stalks projecting from the lens. A disadvantage of contact lens methods is a decreased range of eye movement measurement, usually inappropriate for movements of greater than ± 5 degrees. Also, the cornea slips slightly with respect to the sclera when external forces are applied to it. Stalk methods are very expensive and uncomfortable. There are dangers associated with the fitting of a contact lens with negative pressure, including the possibility of deforming the cornea and causing damage to the accommodation muscles of the eye. Thus, contact lens methods are not practical for most eyetracking applications because they are highly invasive procedures [1,4,7].

Capturing the reflection of a bright light source from the front surface of the cornea is the method used in the class of eye movement instruments known as corneal reflex systems. As with a convex mirror, the reflection of a light source by the cornea forms a virtual image behind the surface that can be imaged and recorded. Figure 5 illustrates the formation of this virtual image. The position of this reflection is a function of eye position. Rotation of the eye about its center produces a relative translation of the image with respect to the pupil or iris, which can subsequently be measured. Corneal reflex techniques are based upon the corneal bulge having a radius of curvature less than that of the eye. Because of the bulge, the corneal reflection moves in the same direction as eye movement. A drawback of the corneal reflex method is that, as a single-feature

optical detection technique, corneal reflex methods have relatively large errors when the detector moves with respect to the head. This inaccuracy arises from the fact that eye translation movements are indistinguishable from eye rotation movements in single feature detection techniques [1]. A 1-mm slip in head position can be equivalent to an eye rotation angle of greater than 12 degrees. This method dictates the use of strict head fixation in order to provide

an acceptable accuracy (± 0.5 degrees). For this reason many head mounted eye movement recording devices require the use of a head strap. It is not necessary, however, to mount the device on a head mounted unit. If head position can be fixed, it is possible to fix a light source in the target field of view. The range of corneal reflex systems that employ a single light source are limited to eye movements of ± 12 -15 degrees vertical or horizontal. Larger movements place the reflex in the nonspherical and peripheral portions of the cornea where it cannot be recorded with most equipment.

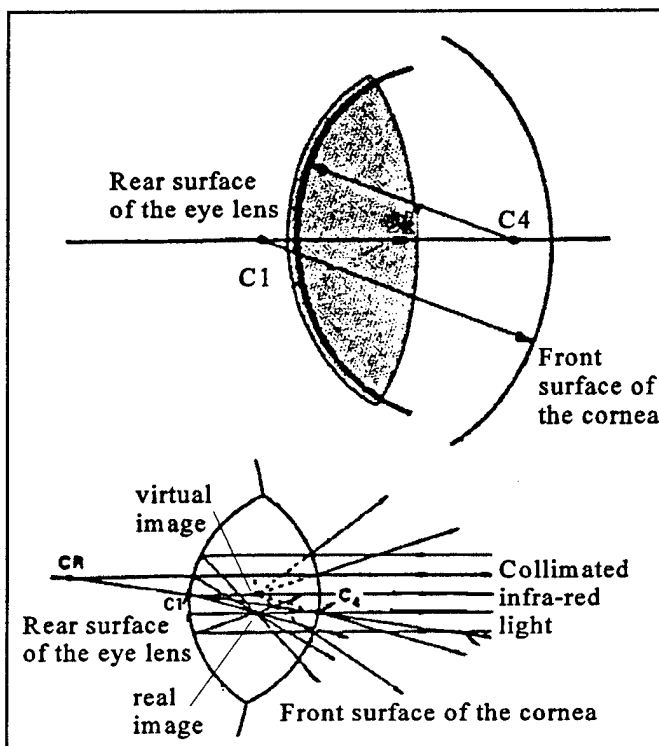


Figure 5. Clam Shell Structure of the Optical Surfaces of the Eye

This figure illustrates the clam shell structure of the front surface of the cornea and the rear surface of the eye lens. The first Purkinje image is the virtual image reflected by the front surface of the cornea. The fourth Purkinje image is the real image reflected by the rear surface of the eye lens. CR-center of rotation of the eye; C1-radius of curvature of the front surface of the cornea; C4-radius of curvature of the rear surface of the eye lens [4].

In addition, the range of recordable movement is limited by the size of the cornea and its occlusion by the eyelids. Variations in cornea shape, tear fluid viscosity and corneal astigmatism are also factors that may limit the accuracy of corneal reflex methods [4,7].

There exist two separate implementations of the point-of-regard method of tracking eye movement. The point-of-regard method is a dual feature technique and unlike the previously discussed methods, the point-of-regard method attempts to measure eye position relative to space, allowing it to distinguish eye rotation from translation. The first implementation of the point-of-regard method involves tracking the corneal reflection center with respect to the pupil center. To accomplish this, the corneal reflection is measured relative to the determined center of the pupil, which is summed with measurements taken from any change in head movement. For a free-head situation, the eye's angle of gaze is calculated to be proportional to the distance between the center of the pupil and the center of the corneal reflection. The corneal reflection method is generally limited by the curvature of the corneal bulge to less than ± 15 degrees. In addition, the pupil is not a stable reference and accuracy is limited to ~ 30 min of arc. The pupil constricts and dilates and does not normally maintain a fixed center with respect to the eyeball, causing the determination of the center of the pupil to be inaccurate. This method lacks the speed necessary to detect some eye movements since television cameras used in this application can operate only as fast as 60 samples/second. Finally, the precision and accuracy are not as good as those that can be obtained with contact lens, limbus tracking, or corneal reflex methods [4,7,8].

An alternate implementation of the point-of-regard method is the dual Purkinje-

image method of tracking eye movement. It involves measuring the corneal reflection relative to a reflection from the eye lens, which is formed from the same infra-red light source used in corneal reflex systems. Every surface of the eye at which there is a change in refractive index results in the occurrence of a reflection. The four reflections present in human eye optics are collectively referred to as Purkinje images.

Figure 6 illustrates the formation of the four images. The second Purkinje image is relatively

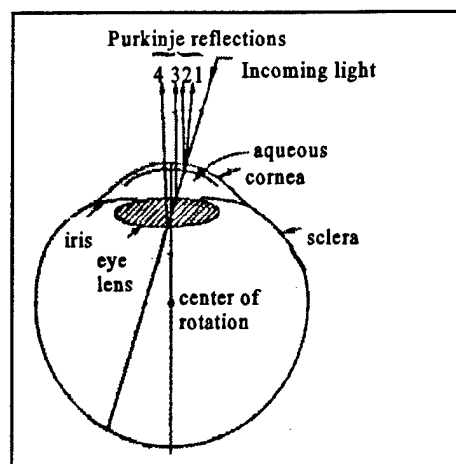


Figure 6. Four Purkinje Images

This is a schematic of the eye illustrating the formation of the four Purkinje images. The first reflection occurs at the front surface of the cornea. The second reflection occurs at the rear surface of the cornea. The third reflection occurs at the front surface of the eye lens. The fourth reflection occurs at the rear surface of the eye lens [9].

dim and like the third Purkinje image, is formed in a plane far from the others. Consequently, these two images are not used. The fourth Purkinje image coincidentally occurs in the same plane as the first Purkinje image, allowing the two images to be focused with the same camera lens. Measurements of the relative displacement between the first and fourth images represent points focused and imaged from planes of different depths in the eye. Therefore, the eye's angle of gaze is proportional to the distance between the first and fourth Purkinje images. Similar to the relationship between the corneal reflection and the pupil center, these two images move together under eye translation but differentially under eye rotation. Changes in separation between these two images is directly related to the angular rotation of the eye and is independent of head translation. The dual Purkinje-image eyetracking method is accurate to 2 minutes of arc in response

to 1 degree of step, making it more precise than the corneal reflex versus pupil center eyetracking method. The eyetracker does fall slightly short of other techniques in field of view (± 15 degrees), which is pupil-diameter dependent, since the fourth Purkinje image is always formed within the pupil [1].

For studies of eye movement control, independent of any other considerations, an eye movement monitoring technique that records eye movement relative to the head is sufficient. However, when the objective is to identifying the elements in a visual scene that are being fixated, the orientation of the eye in space is required. An eye movement monitoring system that records eye movement relative to the head should be used in conjunction with a system that measures head angle and position in order to obtain the orientation of the eye in space. All head measurement techniques require some attachment to the head. The following is a brief discussion of head movement measurement systems that can be integrated with eye movement measurement systems in order to obtain the eye's true point-of-regard in space.

Optical head position sensors can be used to locate the x-y coordinates of a small point attached to the head. By using three targets attached to a helmet (representing motion in the three planes respectively), all translations and rotations of the head can be calculated. The most common optical head tracking devices employ a helmet of some sort with an array of light sources (or photo detectors) on it, and the complimentary devices fixed to the surroundings. A similar system, used in detecting a subject's "line of sight," uses four LEDs mounted on a helmet, each one pulsed at a different frequency. Up to four optical linear position proportional detectors are positioned in the fixed frame

of reference [4].

Three additional types of sensors in the category of head motion detectors include ultrasonic, mechanical, and electromagnetic sensors. Ultrasonic head position sensors permit head location to be sensed by measurement of the distance from source to receiver through the sonic delay time. Mechanical head position sensors can be considered impractical because they require the helmet worn by the subject to be physically attached to a fixed frame of reference in order to determine head movement. Electromagnetic head position sensors require that a search coil be placed on a subject's helmet to detect the angular position of his head. The subject needs to be seated with sensing magnets around him, and a measurement is made of the relative orientation of the search coil with respect to the magnetic field produced by the surrounding system. Inertial measurements of head motion can also be made with miniature gyroscopes and accelerometers mounted on helmets. However, the size and expense of these systems do not currently justify their use [4].

Once a specific method is selected to achieve the purpose of determining head position in space, the problem remains as to how to determine the subject's field of view, then incorporate that information with relative eye position to achieve the final goal of locating the absolute object of fixation. A method for recording head position (linear and angular), or a technique for recording the field of view relative to the head at every sample is required. Mackworth and Mackworth [10] developed a system where a television camera is aligned with the head during free head movements and continuously records the field of view. The accuracy of this system is in the range of ± 2 degrees,

which is relatively poor since it is subject to even greater error caused by the relative movements of the light source attached to a helmet with respect to the eye. Here, the weight of the apparatus is an influential factor; its inertia caused shifting with any rapid movements of the head [4].

In choosing an eyetracking method for this project, the advantages and disadvantages of each method discussed in this section were taken into consideration. A point-of-regard method was preferred to a single feature tracking method. Point-of-regard eyetracking methods are head-movement-independent and would allow the subject to be mobile with the integration of a head tracking system. Large errors due to helmet slippage, associated with single feature tracking methods, would not occur since a point-of-regard method measures eye rotation independent of eye translation. Point-of-regard methods are also non-invasive to the subject. They only require an infra-red light source as opposed to the skin electrodes of electro-oculography or the tight fitting contact lenses of contact lens methods. Of the two point-of-regard eyetracking methods, the dual Purkinje-image method was chosen over the corneal reflex versus pupil center method. The dual Purkinje-image method is the more accurate of the two methods. This inaccuracy is attributed to the fact that the pupil is not a stable reference. Section 3 follows with an in depth discussion of the existing eyetracking system that uses the dual Purkinje-image eyetracking method. It is this existing system that was hoped to be improved upon through work done in this project.

SECTION 3: Existing Dual Purkinje Systems

It is the similarity of the two eye surfaces, the cornea and the eye lens, which makes it possible to track the two Purkinje images simultaneously and formulate a relationship between their relative distance and eye motion. The two surfaces, the front surface of the cornea and the rear surface of the eye lens, resemble a clam shell arrangement; they have approximately the same radius of curvature and have a separation equal to that radius. The effect of collimated light on these nearly spherical surfaces is to produce the two Purkinje images, first and fourth, close to the center and equidistant between the two surfaces [4]. Both images lie almost exactly in the same plane, the pupil plane of the eye. Figure 5 illustrates the clam shell arrangement of the two eye surfaces and the coplanar formation of the two Purkinje images. Even though the fourth Purkinje image is almost the same size and is formed in approximately the same plane as the first Purkinje image, it is very dim because the difference in the refractive index between the eye lens and the vitreous humor is very small. The intensity of the fourth Purkinje image is less than ½% that of the first Purkinje image, making it much more difficult to identify and track compared to the first Purkinje image [9]. The 2-D separation between these two images remains fixed under eye translation but varies with eye rotation.

The only currently known operational dual Purkinje-image eyetracker is the Generation-V dual-Purkinje-image eyetracker. It employs a complex system of optics and servo motors in tracking the first and fourth Purkinje images. Figure 7 is a photograph of the Generation-V dual-Purkinje-image eyetracker. Figure 8 is a simplified schematic of the same system. In this system, an infra-red (IR) light source is directed into the eye causing

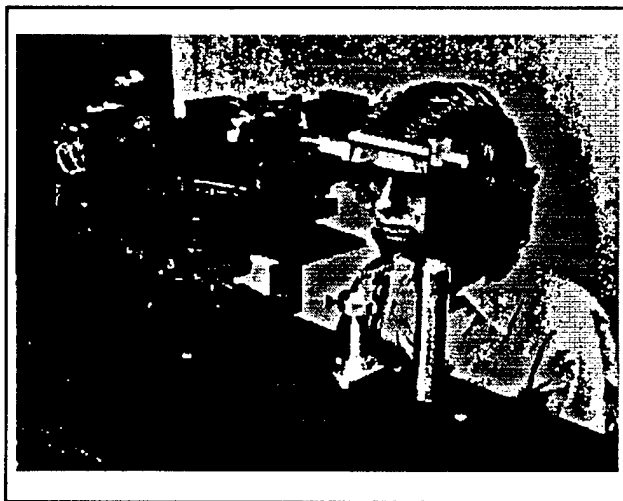


Figure 7. Photograph of the Generation-V Dual Purkinje-Image Eyetracker [1]

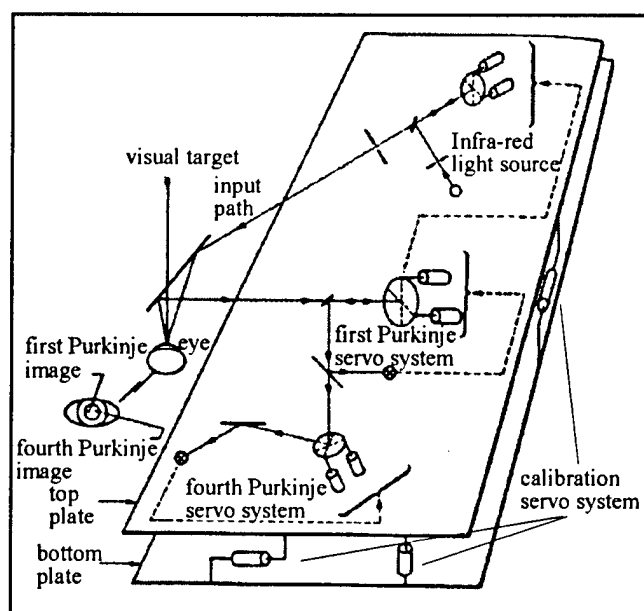


Figure 8. Schematic of the Generation-V Eyetracking System

This simplified schematic illustrates the complexity of this dual Purkinje-image eyetracker [1].

the formation of the two Purkinje images. The first-Purkinje image is an intense, virtual image resulting from input IR reflected at the front surface of the cornea. The fourth Purkinje image is a very weak, real image resulting from input IR reflected from the rear surface of the eye lens. Collection optics (a system of lenses and mirrors) view the eye and focus the two Purkinje images on two photo detectors. Each of the two photo detectors tracks only one of the two images. The photo detectors sense when an image is off-center and send an electrical control signal to the servo motors that rotate a respective mirror in such a way as to cause the image to be continuously

focused on the center of the photo detector. The final output from the photo detectors is the difference between the two electrical signals that are generated for the servo mechanisms

to maintain a centered null condition. This difference is zero if the eye translates. Any difference other than zero can be related to eye rotation [9]. The entire Generation-V eyetracking instrument is mounted on a three-axis motor-driven stage. The requirement for so many optics and servo motors in the system design make it a bulky system, often requiring the subject to remain seated, thereby negating the mobility advantage achieved when using a point-of-regard method of tracking eye movement. The inclusion of the optics and servo motors into the design also make the Generation-V eyetracker a very expensive method of tracking eye movement (approximately \$65,000) [6].

The Generation-V eyetracker has been successfully coupled with a specially designed stimulus deflector system to obtain accurately stabilized images for vision research, and a laser photocoagulator that positions the laser beam on a patient's retina, automatically compensating for the patient's normal eye movements during eye surgery [1].

SECTION 4: Experimental Apparatus

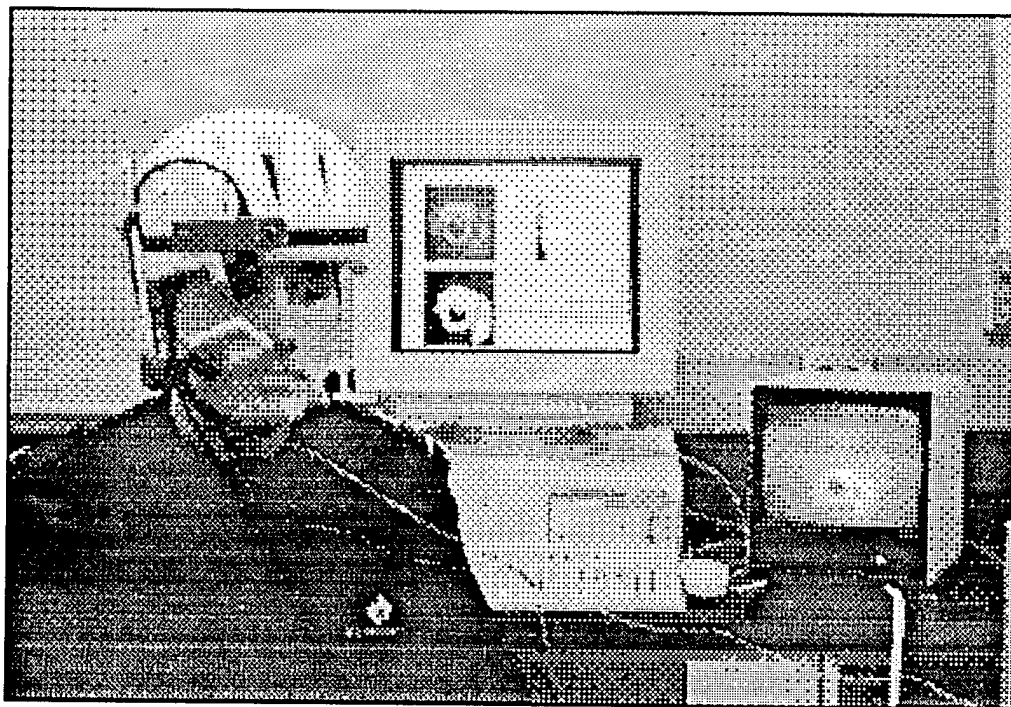


Figure 9. Photograph of the System Designed in this Project

System Design--

Figure 9 is a photograph of the system designed and used in this project. Figure 10 is a block diagram of the system. This system was used in capturing all experimental images of the eye. The major factors influencing system design were desired subject mobility, simplicity in design, and inexpensive cost. The complex design of the Generation-V dual-Purkinje-image eyetracker caused the system to be too bulky to mount on a helmet. Although a head-movement-independent point-of-regard method was used, the Generation-V system design restricted subject mobility. A more lightweight design was accomplished by replacing the complex system of servo motors, mirrors, and photo detectors, used to capture the first and fourth Purkinje images in the

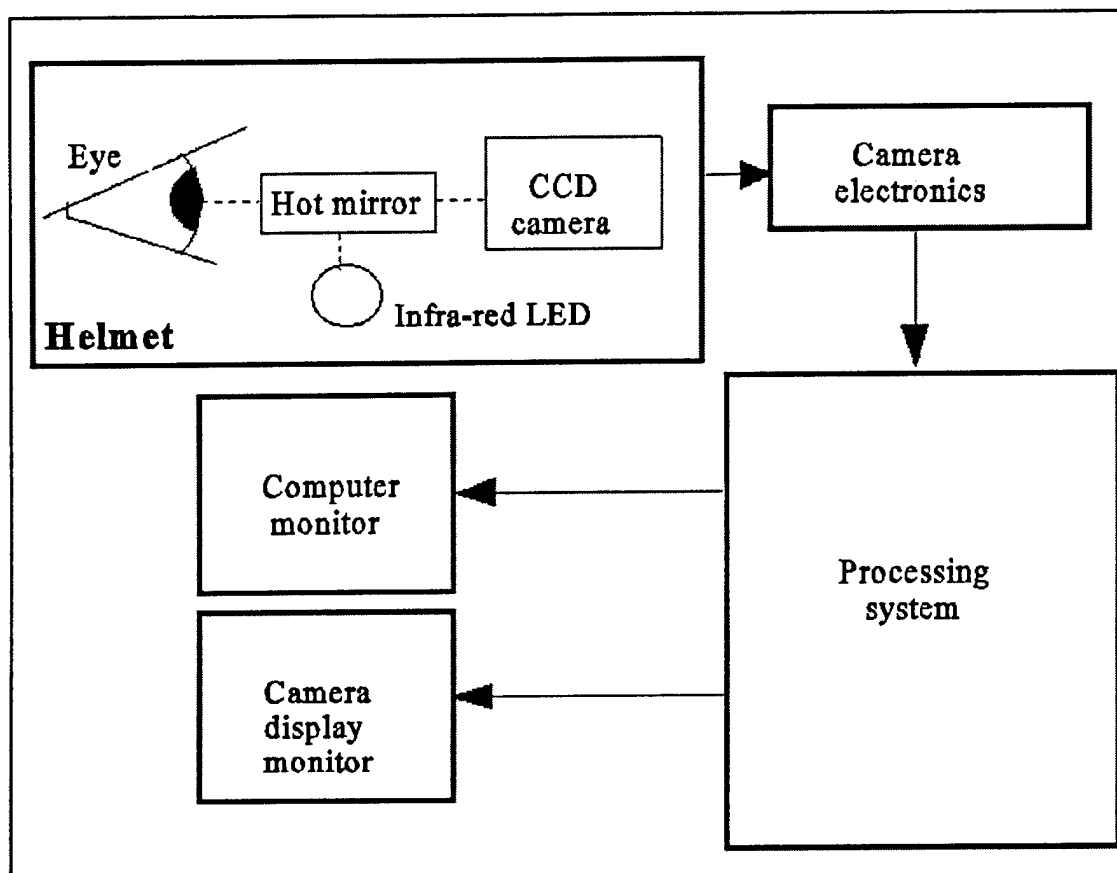


Figure 10. Block Diagram of the Project System

Generation-V system, with a charge coupled device (CCD) camera. As a result of the first and fourth Purkinje images occurring on the same image plane, the pupil plane of the eye, it is possible to capture the images with a focused CCD camera. Recent advances in technology have led to the development of cameras that are extremely small and lightweight. CCD cameras have the added advantage of being sensitive to infra-red light. It is easily possible to mount a small CCD camera on a helmet.

The use of a CCD camera accomplishes the goal of providing subject mobility and reducing complexity in design. Very few other system components were necessary to mount on the helmet to form and capture the Purkinje images. An infra-red (IR) light

source was mounted on the helmet in order to illuminate the eye and cause the four Purkinje images to form. It is possible to obtain an extremely small and lightweight IR source in the form of an IR light emitting diode (LED). In order to ensure proper formation of the four Purkinje images, the IR light was shone through a collimating lens, which causes the resulting IR light to be emitted in parallel rays. To minimize obstruction to a subject's view caused by the IR source and the CCD camera, the two system components were mounted out of the subject's field of view. A single mirror was used to direct the light source into the eye and focus the resulting Purkinje images into the CCD camera. The mirror was placed as much out of the subject's field of view as possible. This was not necessary, however, for a hot mirror was chosen for the system design. A hot mirror reflects infra-red light but is transparent to all visual light. No other system components were required to be mounted on the helmet.

Minimized cost was achieved by using a personal computer equipped with a frame grabber board to process the images captured by the CCD camera. This new system design required the development of image processing algorithms necessary to process the CCD camera images. The development of these algorithms is discussed in detail in Section 5. The existence of reliable feedback was necessary to attack the problem of image processing. Two display systems, the CCD camera monitor and the computer monitor, enabled qualitative analysis of intermediate and final algorithm development. The CCD camera monitor displayed the 640x480 image captured by the CCD camera, unaltered by any image processing algorithms. The computer monitor displayed intermediate and final results in the image processing algorithm. The

combination of these two feedback system components simplified the task of troubleshooting.

One problem with the system design that was discovered in initial experiments was the lack of a mechanism with which to control the intensity of the IR light source. An off-helmet attenuator was added to the system to successfully correct this problem.

The system designed is a monocular tracker. A separate option is binocular tracking. The advantages of a binocular system include extending the total measurable field for eyetracking and increasing the dependability of system measurements by providing redundancy. However, due to the increased complexity, amount of helmet-mounted equipment, processing requirements and associated cost, monocular tracking was chosen over binocular tracking [6].

System Components--

Helmet--The range of eye movement is maximized through system design by keeping the two optical paths, that of the illuminator and the detector/camera, within several degrees of each other. A schematic of the helmet-mounted subsystem is illustrated in Figure 11. In addition, the illuminator and the detector optical axes intersect the eye from near or below the central optical axis, which minimizes the occlusion of the pupil by the upper eyelid. Adjusting the helmet-mounted hot mirror controls the angle at which the optical axes intersect the eye. Therefore, proper adjustment optimizes vertical range of eye movement to be tracked by the eyetracker. It is important that all helmet-mounted components of the system remain out of the subject's field of view in order to achieve minimum

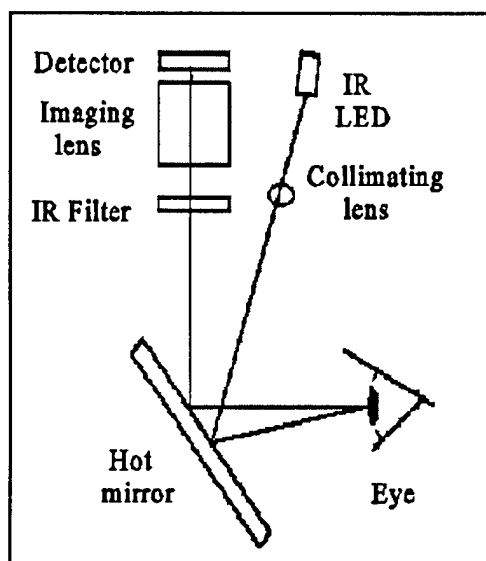


Figure 11. Schematic of the Helmet-Mounted System

This block diagram is of the helmet-mounted system designed in this project.

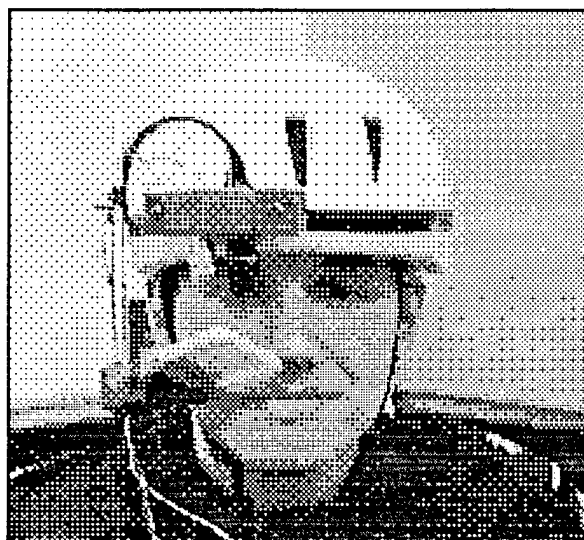


Figure 12. Photograph of the Helmet-Mounted System

In this photograph, the detector, imaging lens, and IR filter are on the right. The IR LED and collimating lens are on the left.

obtrusiveness by the equipment. For this reason, all system components on the helmet are offset to the right side. Figure 12 is a photograph of the helmet.

Camera--A charge coupled device (CCD) camera, operating at 60 Hz, captures 640x480 pixel images of the eye. These full two-dimensional arrays contain 256-level grey scale information.

Processing system--A DT2867 frame grabber board installed in an Intel Pentium Processor based personal computer grabs the images from the CCD camera.

These images are then processed on the personal computer using Borland C++ 4.0.

Feedback system--Two display monitors allow the operator of the system to ensure optimum lighting conditions and correct operation of all image processing algorithms. Two displays also greatly aided in the development of image

processing algorithms by allowing the operator to view the images before and after processing.

SECTION 5: Image Processing Algorithms

Objective--The main objective of this project is to determine the feasibility of using image processing techniques to locate and track the first and fourth Purkinje images within eye images captured by a CCD camera. Accomplishing this objective would result in the successful design of a lightweight and inexpensive dual Purkinje-image eyetracker.

To meet this objective, it was necessary to conduct experiments by capturing the images of the eyes of several different subjects. Images were captured from ten subjects in various lighting conditions. The subjects were instructed to look in several different directions. In analyzing the images taken during these experiments, certain discoveries were made, essential in the development of the image processing algorithms. It became apparent that in every lighting condition and angle of eye rotation, the first Purkinje image would be the brightest object in an eye image. This fact, however, was limited to instances where the formation of the first Purkinje image occurred. For large eye angles, the IR light source did not shine on the cornea and as a result, the first Purkinje image did not form. In addition, the experimental images verified the researched fact that the fourth Purkinje image always appears within the pupil. This fact led to the decision to define the area of the pupil as the search area for the fourth Purkinje image. The pupil was discovered to be homogenous in grey level with the exception of the fourth Purkinje image allowing the search for the dim fourth Purkinje image to be an achievable task. The following is a more detailed description of the development of the image processing algorithms generated in this project.

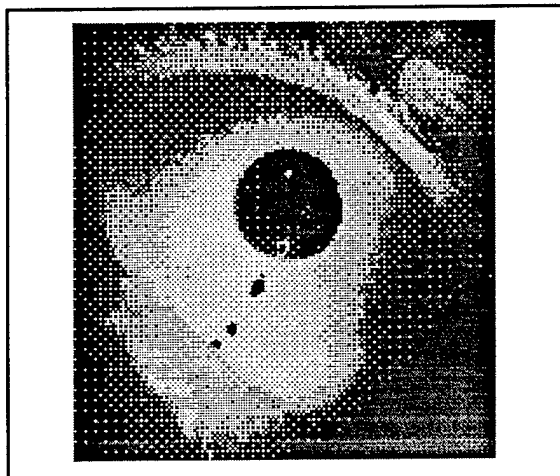


Figure 13. Photograph of Eye Glints

This figure illustrates the occurrence of unwanted eye glints in a 256x256 pixel image of the eye.

Fortunately, this image was still centered around the first Purkinje image. Lowering the intensity of the IR light source greatly aids in eliminating the occurrence of most eye glints.

The first step in the attempt to successfully accomplish the project's objective was to capture several different images of the eye to determine what features in an eye image could be identified as critical to the task of determining Purkinje image location.

Initial experiments revealed that the first Purkinje image is always the brightest spot within an eye image. In an

image of 256 grey levels, zero representing pure black and 255 representing pure white with levels in between representing varying levels of grey, the first Purkinje image always has a value of 255.

The frame grabber always acquires a 640x480 pixel image, but the eye within the image never fills up the entire 640x480 pixel frame. In an effort to speed processing, a 256x256 square centered around the bright first Purkinje image, was cropped from the 640x480 pixel frame. This cropping process is a successful technique since all vital image information can be found close to the first Purkinje image, which was discovered always to be near the pupil. This 256x256 pixel image is then displayed on the computer monitor allowing the operator to ensure that the entire pupil is within this 2-D array.

One problem that initially arose in the search for the first Purkinje image was that glints often occur within the eye boundary. These glints occur due to the shape of the

reflecting surface and the illuminator beam angle of incidence; tear film over the cornea and sclera also often produces very bright glints. Figure 13 is a photograph of an eye containing unwanted occurrences of eye glints. These glints often have a grey level of, or near 255 thereby making them indistinguishable from the first Purkinje image. Proper adjustment of the IR light source can remove most unwanted glints. This was accomplished by lowering the intensity of the IR LED to a level where only the first Purkinje image was bright enough to have a grey level of 255. The glints still occurred, however, they now occurred at a lower grey level. To aid the operator in identifying the features in the eye, all pixels of grey level 255 were changed to the color red.

Once the first Purkinje image was successfully located and the image to be processed was successfully reduced, the problem remained to locate the fourth Purkinje image. The initial difficulty in the search for the fourth Purkinje image was that it is very small, four or five pixels in size and it occurs at an indistinctive grey level. Unlike the search for the first Purkinje image, it was not possible simply to search through the image for the specific grey level because many separate features of the eye occur at the same grey level. In fact, experimentation revealed that the majority of the iris was of the same grey level as the fourth Purkinje image. Research conducted by Crane and Steele [1] indicated that the fourth Purkinje image is always found within the boundary of the pupil. Through experimentation this occurrence was verified. By narrowing the search area to the area of the pupil, it appeared possible to locate the fourth Purkinje image, which occurred as a noticeably bright feature in comparison to the 'black' of the pupil.

Isolation of the pupil in order to define a search area became the next image

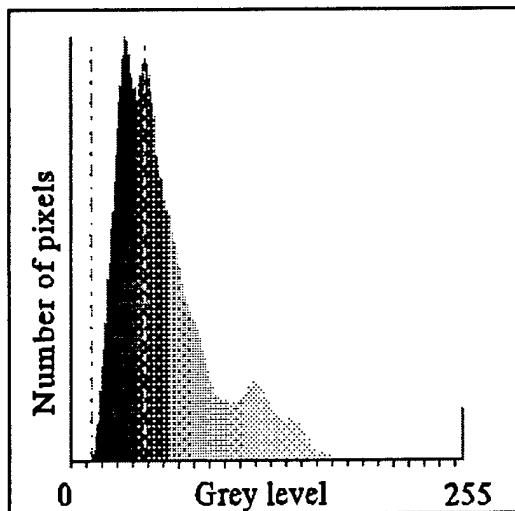


Figure 14. Histogram of an Eye Image
This is a plot of grey level versus number of pixels at that grey level within a 256x256 pixel image of an eye.

processing goal. Initial attempts were made to isolate the pupil by setting all values near a grey level of zero to a single black level and the remaining values of grey to a single value of white. Thresholding the image using one number, setting all grey levels below the one number to a single black level and all grey levels above the one number to a single white level, proved

unsuccessful. It was discovered that although the pupil was composed of homogenous grey levels, these levels were closer to the grey level of approximately 30 than to the grey level of zero, for the lighting conditions used in the experiments conducted in this project. Essentially, the pupil is not black in color but rather a very dark shade of grey. In order to isolate the pupil, the range of grey levels that are contained within the pupil had to be determined. Once this range of values could be established then a thresholding process could be applied to the image of the eye and the pupil could be successfully isolated. All grey level values found within the pupil would be set to black and all values outside this range would be set to white by the thresholding procedure.

The image processing technique of histogramming aided in the task of determining the range of grey level values that are located within the pupil. A histogram is a plot of grey level versus number of pixels at that grey level within the image. Figure

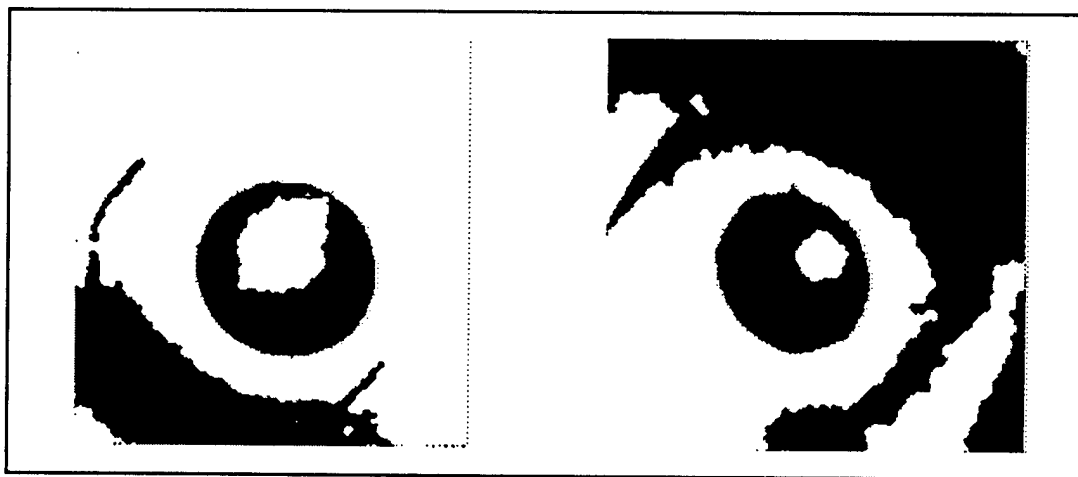


Figure 15. Successfully Binarized Eye Images

The correct selection of the thresholding range to be used in binarizing a 256x256 pixel image of the eye results in the successful isolation of the pupil. The pupil area is defined as the search area for the fourth Purkinje reflection.

14 is the plot of a histogram of one image of the eye. With histogramming it becomes

possible to separate features in an image that have homogeneous grey level structure.

Through experimentation, it was discovered that the first peak in the plot of the

histogram represents the average grey level value found within the pupil of most subjects

seen under similar conditions. A range of 40 pixels centered around this average value

was used in thresholding the image. Figure 15 illustrates two images where the

histogram method was used to successfully binarize the images. A properly binarized

image, resulting in successful isolation of the pupil aided in the following task of defining

the pupil as the search area for the fourth Purkinje image.

For feedback purposes, the histogram and the resulting binary image were presented on a monitor. This allowed the operator to see that the pupil was being properly isolated. It also ensured that the proper amount of light was correctly set. If there was too much infrared light being shone in a subject's eye, the pupil would appear too bright and the pupil grey levels would not occur in the first peak, resulting in white



Figure 16. Results of a Majority Filter

A majority filter creates a white pixel if five or more pixels in a 3x3 area are white; otherwise, it sets the pixel to black. This filter is useful for filling small holes or for smoothing rough edges.

pixels in the binary plot that are indistinguishable from the white pixels of the iris and the sclera. If not enough infra-red light was being shone in a subject's eye, parts of the iris would appear as dark as the pupil, again causing the pupil to be improperly isolated from the rest of the image. A properly binarized image greatly facilitates the task of locating the pupil.

The next task was to properly outline (contour) the pupil. Once the image was binarized, a majority filter was passed over the binarized image in order to smooth out the edges and therefore make the contouring part of the process an easier task. Binarized images often have ragged edges at the boundary between the pupil and the iris. Since the boundary contains grey levels that describe the pupil in addition to grey levels that describe the iris, the edge that results is a mix of scattered black and white pixels. A majority filter counts up the number of pixels in a 3x3 array that are white and the number that are black, if the majority are white, the center pixel of the array is replaced by a white pixel, and if the majority are black, by a black pixel. This filter was passed

over the entire array and interpolated (smooths) the rough edges. Figure 16 illustrates a properly binarized image before and after it is passed over by a majority filter. The eye images in Figure 15 were smoothed using a majority filter.

By contouring the pupil, the task of defining the search area for the fourth Purkinje image was accomplished. As mentioned above, due to the cropping procedure the first Purkinje image is located in the center of the 256x256 image. Through experimentation on the aforementioned ten subjects, it was discovered that the first Purkinje image always occurs in close proximity to, or within the pupil. Therefore, in the algorithm to contour the pupil the white pixels of the first Purkinje image were used as a starting point. The algorithm searches first to the right for a black pixel, if none is found within the first 100 pixels, then the search continues down, left, and finally up. Upon locating the first black pixel, the search proceeds in the same direction until a white pixel is located. The second edge then becomes the starting point of the pupil contour. This step is necessary because if the first Purkinje image occurs entirely within the pupil, the first edge encountered would not be the outside edge of the pupil, but instead the edge of the image occurring within the pupil. Once an edge of the pupil is located, a simple contouring algorithm follows around the entire edge of the pupil until the initial starting point is located. This contour is presented on the monitor, overlaying in yellow the binarized image of the eye, again enabling the operator to ensure correct execution of the algorithm. Two arrays were set up, one to store the pixel coordinates of the left edge of the pupil and the other to store the coordinates of the right edge of the pupil, for each row of pixels. With the search area defined by these arrays, it was possible to search the

area of the pupil in an original (non-binarized) image of the eye for the location of the fourth Purkinje image.

In order to ensure that the first Purkinje image is not mistaken for the fourth Purkinje image, all grey level values above 128 are ignored in the final search for the fourth Purkinje image within the pupil area. Searching only the grey levels between zero and 128, the location of the maximum grey level value within this range was discovered to be the location of the fourth Purkinje image. This pixel is then plotted as a yellow pixel both on the binarized image and the non-binarized image of the eye. The operator uses this feedback to ensure that the fourth Purkinje image was accurately located. Figure 17 is an illustration of successfully processed images, which resulted in the accurate location of the first and fourth Purkinje images. The Appendix contains the Borland C++ 4.0 source code written for the development of these algorithms.

With the location of the first and fourth Purkinje images determined, it now becomes possible to correlate the distance and orientation between the two images to eye angle of rotation. In order to accomplish this task, it is necessary to calibrate the eyetracking system for use with each subject. A subject is instructed to fixate on predetermined points on a computer screen without moving his head. Once the distance and orientation between the two Purkinje images is calculated for each presented screen location, all other locations on the screen can be interpolated for subsequent images of the eye. This specific determination of the point of eye fixation refers back to the application of using an eyetracker to serve as an eye guided mouse for use with a computer. Further mathematics would be applied to determine eye angle of rotation.

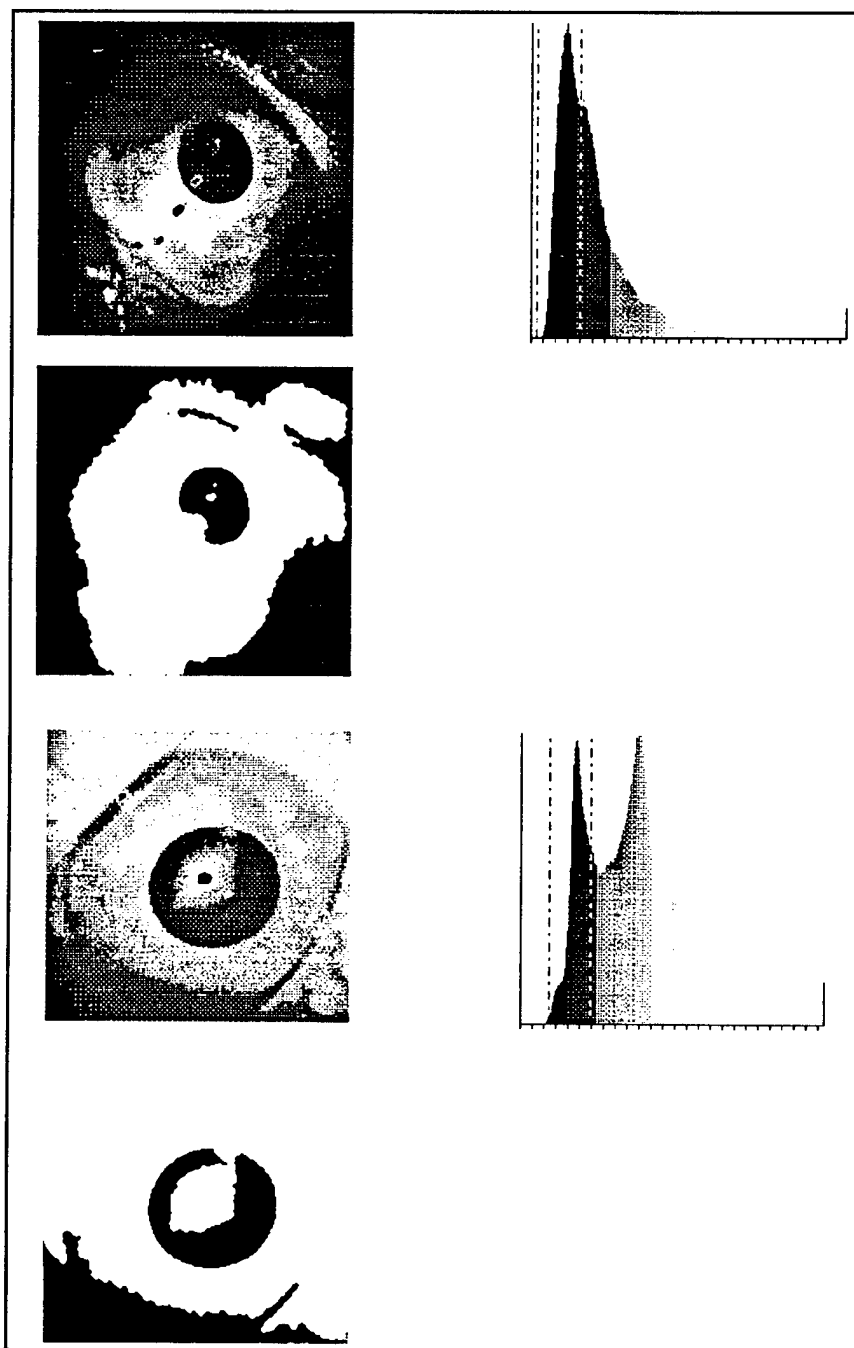


Figure 17. Two Successfully Processed Images

This figure illustrates two instances where a 256x256 pixel image was successfully centered around the first Purkinje image. Next a histogram was plotted and a thresholding range was selected for the binarization of the eye images. The successfully binarized eye images are also shown. From these images, successful contours of the pupil were made within which the fourth Purkinje image was accurately located.

These calculations would take into consideration the distance between the subject and the object of fixation.

SECTION 6: System Performance

Feedback was essential in qualitatively evaluating system performance of the eyetracker. Two displays were used for feedback. One display is of the video eye image captured by the detector. The second display is of the smaller 256x256 pixel image, the plot of the histogram, and the binarized image of the eye used in isolating the pupil. Throughout the development of the image processing algorithms, the two displays served as a self-checking tool. As a result, many of the initial problems with the algorithms were solved.

The initial problems included eliminating unwanted glints occurring on the surface of the eye. These glints could be mistaken for the first Purkinje reflection. The problem was solved by reducing the intensity of the light source. Proper adjustment of the light source resulted in more uniform illumination of the pupil of the eye and improved binarizing of the eye images. The system designed in this project is very sensitive to the amount of infra-red light being used. It was also recognized early on that it was necessary to have the focal point of the light source collimating lens on the pupil plane of the eye. This ensures the formation of detectable first and fourth Purkinje reflections.

Other initial difficulties were discovered when eye images were taken from several different subjects. Initially the eye images were binarized using a set range of pixel grey levels. It was discovered that different subjects had different ranges of pixel grey levels that defined the pupil area. The image processing algorithm was then adjusted to automatically binarize the image after defining the pupil area, which include

the grey levels found in the first peak of the histogram, regardless of where the first peak occurred.

Some problems still exist in the image processing algorithms that need to be worked out. If the first and fourth reflections are located near each other within an image of the eye, the bright first reflection will obscure the fourth reflection. Another problem stems from the fact that the system should be able to process images of the eye during the worst case situation of maximum eye rotation. The eye can rotate ± 75 degrees whereas for the fourth Purkinje reflection to be formed the eye must be restricted to ± 15 degrees. If the pupil is constricted this range is even smaller.

The infra-red light source used in this project does not emit over a large enough area to ensure that the pupil remains fully illuminated during larger angles of eye rotation. When the pupil is not fully illuminated, it blends in to the rest of the non-illuminated eye. As a result, proper binarization of the 256x256 pixel image does not occur and the search area of the pupil is not properly defined. In these situations, the fourth Purkinje reflection cannot be located accurately.

The importance of proper infra-red illumination cannot be stressed enough. It is apparent now that many problems initially discovered would have been solved more efficiently if they had been attacked by hardware system design as opposed to software image processing. More work is needed to optimize the hardware design. The feasibility of using image processing in this project can now justify the effort that will be required in perfecting the hardware system design.

SECTION 7: Conclusion

In this project, image processing algorithms were developed that accurately locate the first and fourth Purkinje images within images of the eye captured by a CCD camera. The system designed to capture these images is comprised of an infra-red LED light source, a hot mirror, and a CCD camera mounted on a helmet. The helmet-mounted system is connected to camera electronics, a personal computer processing system, and two visual displays.

The image processing algorithms that were designed locate the first Purkinje image by searching an eye image for the brightest pixel. Next, the pupil is isolated through the use of a histogramming technique. Once the pupil is isolated, the search area for the fourth Purkinje image is defined. Finally, the pupil is searched for the location of the fourth Purkinje image. The 2-D locations of these two Purkinje images may be used to determine eye rotation in two planes.

The feasibility of using image processing techniques to locate and track the first and fourth Purkinje images has been demonstrated. This allows for the design of a lightweight and inexpensive eyetracking system that can be suitable for a wide variety of applications that require accurate knowledge of eye position.

SECTION 8: Future Activities

Future work in this area can be done to develop a system that continuously tracks the first and fourth Purkinje images and relates them to eye position. There is the potential to develop a real-time eyetracking system using the processing algorithms developed in this project. Future work also needs to be done in determining the accuracy, precision, and stability of this system to see how it compares to the existing Generation-V dual-Purkinje image eyetracker. Once enough research is conducted to verify the reliability of the techniques developed in this project, future activities include integrating this system with the eyetracking applications mentioned in Section 1 of this paper.

REFERENCES

1. Crane, H.D. and C.M. Steele. *Generation-V Dual-Purkinje-Image Eyetracker*. Applied Optics Volume 24, 527 (1985).
2. Yamada, M. and T. Fukuda. *Eye Word Processor (EWP) and Peripheral Controller for the ALS Patient*. IEE Proceedings Volume 134, 328 (1987).
3. Fendrich, R. *Durip-Improved Eye Movement Monitoring Capabilities for Studies in Visual Cognition*. Alexandria, Virginia: Defense Technical Information Center (1990).
4. Young, L.R. and D. Sheena. *Survey of Eye Movement Recording Methods*. Behavior Research Methods & Instrumentation Volume 7, 397 (1975).
5. Pillalamarri, R.S., B.D. Barnette, R. Karsh, and D.P. Birkmire. *Program "Cluster"-An Identification of Fixation Cluster Characteristics*. Alexandria, Virginia: Defense Technical Information Center (1992).
6. Borah, J. *Helmet Mounted Eye Tracking for Virtual Panoramic Display Systems--Volume 2: Eyetracker Specification and Design Approach*. Alexandria, Virginia: Defense Technical Information Center (1989).
7. Borah, J. *Helmet Mounted Eye Tracking for Virtual Panoramic Display Systems--Volume 1: Review of Current Eye Movement Measurement Technology*. Alexandria, Virginia: Defense Technical Information Center (1989).
8. Sheena, D. *Patter-Recognition Techniques for Extraction of Features of the Eye from a Conventional Television Scan*. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers (1976).
9. Crane, H.D. and C.M. Steele. *Accurate Three-Dimensional Eyetracker*. Applied Optics Volume 17, 691 (1978).
10. Mackworth, N.F. and H.H. Mackworth. *Eye Fixations Recorded on Changing Visual Scenes by the Television Eye-Marker*. Journal of the Optical Society of America Volume 48, 439 (1958).

BIBLIOGRAPHY

Bar-Shalom Y., Shertukde and Pattipati. *Measurements from an Imaging Sensor for Precision Target Tracking*. IEEE Transactions on Aerospace and Electronic Systems Volume 25, 863 (1989).

Borah, J. *Helmet Mounted Eye Tracking for Virtual Panoramic Display Systems--Volume 1: Review of Current Eye Movement Measurement Technology*. Alexandria, Virginia: Defense Technical Information Center (1989).

Borah, J. *Helmet Mounted Eye Tracking for Virtual Panoramic Display Systems--Volume 2: Eyetracker Specification and Design Approach*. Alexandria, Virginia: Defense Technical Information Center (1989).

Crane, H.D. and C.M. Steele. *Accurate Three-Dimensional Eyetracker*. Applied Optics Volume 17, 691 (1978).

Crane, H.D. and C.M. Steele. *Generation-V Dual-Purkinje-Image Eyetracker*. Applied Optics Volume 24, 527 (1985).

Fendrich, R. *Durip-Improved Eye Movement Monitoring Capabilities for Studies in Visual Cognition*. Alexandria, Virginia: Defense Technical Information Center (1990).

Fu, K.S., R.C. Gonzalez and C.S.G. Lee. ROBOTICS: Control, Sensing, Vision, and Intelligence. New York, NY: McGraw-Hill Publishing Company (1987).

Gonzalez, R.C. and P. Wintz. DIGITAL IMAGE PROCESSING. Reading, Massachusetts: Addison-Wesley Publishing Company (1977).

Hutchinson, T.E., K.P. White, Jr., W.N. Martin, K.C. Reichert, and L.A. Frey. *Human-Computer Interaction Using Eye-Gaze Input*. IEEE Transactions on Systems, Man, and Cybernetics Volume 19, 1527 (1989).

Jacob, R.J.K. *What You Look At is What You Get: The Use of Eye Movements in Human-Computer Interaction Techniques*. Washington, D.C.: Naval Research Center (1990).

Li, H.F., F.H.Y. Chan, P.W.F. Poon, W.F. Yan, W.Y. Wong and J.C. Hwang. *Real-Time On-Line Pattern Recognition of Eye Position and Movement*. IEE Proceedings Volume 132, 293 (1985).

Mackworth, N.F. and H.H. Mackworth. *Eye Fixations Recorded on Changing Visual Scenes by the Television Eye-Marker*. Journal of the Optical Society of America

Volume 48, 439 (1958).

Merchant, J., R. Morrisette and J.L. Porterfield. *Remote Measurement of Eye Direction Allowing Subject Motion Over One Cubic Foot of Space*. IEEE Transactions on Biomedical Engineering (1974).

Milbocker, M.T. and Feke, G.T. *Intensified Charge-Coupled-Device-Based Eyetracker and Image Stabilizer*. Applied Optics Volume 31, 3719 (1992).

Nevatia, R. MACHINE PERCEPTION. Englewood Cliffs, NJ: Prentice-Hall, Inc. (1982).

Pillalamarri, R.S., B.D. Barnette, R. Karsh, and D.P. Birkmire. *Program "Cluster"-An Identification of Fixation Cluster Characteristics*. Alexandria, Virginia: Defense Technical Information Center (1992).

Rinard, G.A., R.W. Matteson, R.W. Quine and R.S. Tegtmeyer. *An Infrared System for Determining Ocular Position*. ISA Transactions Volume 19, 3 (1980).

Sheena, D. *Pattern-Recognition Techniques for Extraction of Features of the Eye from a Conventional Television Scan*. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers (1976).

Sheena, D. and J. Borah. *Compensation for Some Second Order Effects to Improve Eye Position Measurements*. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers (1981).

Thorisson, K.R. and D.B. Koons. *Unconstrained Eye Tracking in Multi-Modal Natural Dialogue*. Cambridge, MA: MIT Media Laboratory (1991).

Tole, J.R. and L.R. Young. *Digital Filters for Saccade and Fixation Detection*. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers (1981).

Yamada, M. and T. Fukuda. *Eye Word Processor (EWP) and Peripheral Controller for the ALS Patient*. IEE Proceedings Volume 134, 328 (1987).

Young, L.R. and D. Sheena. *Survey of Eye Movement Recording Methods*. Behavior Research Methods & Instrumentation Volume 7, 397 (1975).

SOURCE CODE

***A.C.Chamberlain, Trident Project, Dual Purkinje-Image Eyetracker:**

***** This program contains the source code for the image processing algorithms developed in this project. They include: cropping the 640x480 pixel image to a 256x256 pixel image centered around the bright first Purkinje reflection, calculating the image histogram, automatically defining the 40 grey level area by which the image is to be binarized, binarizing the image, 'smoothing' the image with a majority filter, contouring the pupil area in the binarized image, and searching the pupil area of the grey level 256x256 pixel image for the occurrence of the fourth Purkinje reflection.

***setting up the initial conditions**

***** opening up the necessary libraries to run the program in Borland C++ 4.0

***** defining the functions necessary to operating in a windows graphics environment

```
#include <owl\applicat.h>
#include <owl\framewin.h>
#include <owl\gdiobjec.h>
#include <string.h>
#include <owl\dc.h>
#include <owl\color.h>
#include <owl\point.h>
#include <stdio.h>
#include <owl\scrollba.h>
#include <owl\button.h>
#include <owl\opensave.h>
#include <dir.h>
#include <math.h>
#include <dos.h>
#define DT2867 1
#include "c:\BC4\DT_IDT_I.H"
#define CM_FILEEXIT 24338
#define CM_FILEOPEN 24332
#define CM_GRAB 501
#define CM_CONVERT 24339
#define num_rows 256
int col=0;
```

***sets up all the public variables and functions**

```
class TMyWindow : public TWindow
{
    public:
        TMyWindow(TWindow * parent=0);
        ~TMyWindow();
        void Paint(TDC&, BOOL, TRect&);
        TPalette* Tpal;
        unsigned char **ia,**ib;
        unsigned long hist[256];
        int peak;
        FILE *fp,*outf;
        BOOL EFlag;
        BOOL IFlag;
        void SetupWindow();
        void Open();
```

```

        void Quit();
        void Grab();
        void Convert();
        void Delay(int dly);
        void Init_2867();
        void Live_image(int camera,int ilut);
        void Freeze_image(int camera,int ilut, int plut, int buffer);
        void Transfer_image(void);
        DECLARE_RESPONSE_TABLE(TMyWindow);
};

```

***defines the commands used in operating the graphics window in this project**

*** grab is a defined function which captures a 640x480 pixel image from the CCD camera**
*** convert is a defined function which converts a tiff graphics file to an img graphics file, this**
*** command was necessary initially when the images were captured from an outside image**
*** processing software program: Global Lab**

```

DEFINE_RESPONSE_TABLE1(TMyWindow, TWindow)
    EV_COMMAND(CM_FILEEXIT,Quit),
    EV_COMMAND(CM_FILEOPEN,Open),
    EV_COMMAND(CM_GRAB,Grab),
    EV_COMMAND(CM_CONVERT,Convert),
END_RESPONSE_TABLE;

```

```

TMyWindow::TMyWindow(TWindow *parent)
{
    Init(parent,0,0);
    TClientDC tdc(*this);
    IFlag = FALSE;
}

```

***The next couple of lines of code sets up the palette to be used in displaying the cropped 256x256**
***pixel image and the plotted histogram. The lowest 32 levels of grey are to be varying shades of**
***green. The highest two levels of grey are to be colored red (the is to aid the user in rapidly**
***identifying if the bright first Purkinje reflection has been properly located). The remaining levels of**
***grey are left in varying shades of grey.**

```

void TMyWindow::SetupWindow(void)
{
    int i;
    EFlag = FALSE;
    PALETTEENTRY pe[192];
    TWindow::SetupWindow();
    for(i=64;i<96;i++)
    {
        pe[i].peRed = 0;
        pe[i].peGreen = (i-64)*8;
        pe[i].peBlue = 0;
        pe[i].peFlags = 0;
    }
    for(i=96;i<190;i++)
    {
        pe[i].peRed = (i-64)*2;
        pe[i].peGreen = (i-64)*2;
    }
}

```

```

        pe[i].peBlue = (i-64)*2;
        pe[i].peFlags = 0;
    }
    for(i=190;i<192;i++)
    {
        pe[i].peRed = (i-64)*2;
        pe[i].peGreen = 0;
        pe[i].peBlue = 0;
        pe[i].peFlags = 0;
    }
    Tpal = new TPalette(pe,192);

```

***Checks to make sure there is sufficient memory to contain the grey levels of each of the pixels in the *256x256 image array**

```

    if(!EFlag) ib = new unsigned char *[num_rows];
    if (!EFlag && (ib == 0))
    {
        MessageBox("Cannot allocate enough memory for
image","WARNING",MB_OK
                                                    |MB_ICONSTOP);
        EFlag = TRUE;
    }
    else for (int y=0; (y<num_rows) && (!EFlag); y++)
    {
        ib[y] = new unsigned char [256];
        if (ib[y] == 0)
        {
            MessageBox("Cannot allocate enough memory for
image","WARNING",MB_OK
                                                    |MB_ICONSTOP);
            EFlag = TRUE;
            break;
        }
    }
    if(!EFlag) ia = new unsigned char *[num_rows];
    if (!EFlag && (ia == 0))
    {
        MessageBox("Cannot allocate enough memory for
image","WARNING",MB_OK
                                                    |MB_ICONSTOP);
        EFlag = TRUE;
    }
    else for (int y=0; (y<num_rows) && (!EFlag); y++)
    {
        ia[y] = new unsigned char [256];
        if (ia[y] == 0)
        {
            MessageBox("Cannot allocate enough memory for
image","WARNING",MB_OK
                                                    |MB_ICONSTOP);
            EFlag = TRUE;
            break;
        }
    }

```

```

    }
}
Init_2867();
}

```

***instructs the computer to quit the application if the function is opted by the user**

```

void TMyWindow::Quit(void)
{
    Destroy();
}

```

***instructs the computer to open a dialog box in order to allow the user to open an img graphics file
 *containing a previously captured and cropped 256x256 pixel image. The image is then allocated
 *memory in a two-dimensional array, displayed in the open graphics window, and a histogram of the
 *image is also calculated and displayed in the open graphics window.**

```

void TMyWindow::Open(void)
{
    int x,y,sum,i,j;
    unsigned char pix;
    TfileOpenDialog::TData FilenameData(OFN_FILEMUSTEXIST | OFN_HIDEREADONLY
                                         | OFN_PATHMUSTEXIST, "Image Files
(*.img) |*.img|",0,
                                         "", "img");
    if(TFileOpenDialog(this,FilenameData).Execute() != IDOK)
    {
        MessageBox("Cannot Open File","WARNING",MB_OK | MB_ICONSTOP);
        EFlag = TRUE;
    }
    else (fp=fopen(FilenameData.FileName,"rb"));
    if(fp == NULL)
    {
        MessageBox("Cannot Open File","WARNING",MB_OK | MB_ICONSTOP);
        EFlag = TRUE;
    }
    for(y=0;(y< num_rows) && (!EFlag); y++)
    if(fread(ib[y],1,256,fp) < 256)
    {
        MessageBox("Cannot read image","WARNING",MB_OK
                  |MB_ICONSTOP);
        EFlag = TRUE;
    }
    for(y=0;y< num_rows;y++)
    {
        hist[y]=0;
    }
    for(y=0;y< num_rows;y++)
    {
        for(x=0;x < num_rows;x++)
        {
            pix = *(ib[y]+x);
            hist[pix]++;
        }
    }
}

```

```

    }
    for(i=0;i<248;i++)
    {
        sum=0;
        for(j=i;j<(i+8);j++)
            sum+=hist[j];
        sum >>= 3;
        hist[i]= sum;
    }
    if(!EFlag)
    {
        EFlag = TRUE;
        Invalidate(TRUE);
    }
}

```

***this function delays the processing in 10 msec increments every time it is called**

```

void TMyWindow::Delay(int dly)
{
    struct dostime_t dtime;
    int i,t0;
    for(i=0;i< dly;i++)
    {
        _dos_gettime(&dtime);
        t0=dtime.hsecond;
        while(dtime.hsecond == t0)
        {
            _dos_gettime(&dtime);
        }
    }
}

```

***The following function initialized the framegrabber board used to capture the 640x480 CCD camera images. The delay calls ensure that the board will initialize properly. Functions included in this section set up the frame grabber to display a live image on a camera display, freeze the image, capture the image as a 640x480 grey level array, transfer the image to the Borland C++ working environment, crop the image to a 256x256 pixel image centered around the bright first Purkinje reflection, calculate a histogram based on the 256x256 image and display the image and histogram in the open graphics window. These functions all run when an image is grabbed immediately from the framegrabber board without going through outside processing software or opening a previously saved img graphics file.**

```

void TMyWindow::Init_2867(void)
{
    -----
    int i, j, ilut, plut;
    outport (DT2867_BASE, DEF_0|DISBUF1);
    Delay (1);
    outport (DT2867_BASE+4, DEF_4);
    Delay (1);
    outport (DT2867_BASE+1044, DEF_1044);
    Delay (1);
    outport (DT2867_BASE+1046, 1883);
}

```

```

Delay (1);
outport (DT2867_BASE+1048, 2816);
Delay (1);
for (ilut=0; ilut<8; ilut++) {
    outport (DT2867_BASE+1044, DEF_1044|(ilut<<8));
    outport (DT2867_BASE+1040, 0);
    Delay (1);
    switch (ilut) {
        case 0:
            for (i=0; i<256; i++)
                outport (DT2867_BASE+1042, 1);
            break;
        case 1:
            for (i=0; i<256; i++)
                outport (DT2867_BASE+1042, 255-i);
            break;
        case 2:
            for (i=0; i<256; i++)
                outport (DT2867_BASE+1042, (i>64)?255:0);
            break;
        case 3:
            for (i=0; i<256; i++)
                outport (DT2867_BASE+1042, (i>128)?255:0);
            break;
        case 4:
            for (i=0; i<256; i++)
                outport (DT2867_BASE+1042, (i>192)?255:0);
            break;
        case 5:
            for (i=0; i<256; i++)
                outport (DT2867_BASE+1042, (i&0x40)?255:0);
            break;
        case 6:
            for (i=0; i<256; i++)
                outport (DT2867_BASE+1042, (i&0x20)?255:0);
            break;
        case 7:
            for (i=0; i<256; i++)
                outport (DT2867_BASE+1042, (i&0x10)?255:0);
            break;
        default:
            break;
    }
}
for (plut=0; plut<8; plut++) {
    outport (DT2867_BASE+2058, DEF_2058|(plut<<12));
    outport (DT2867_BASE+2060, 0);
    Delay (1);
    for (i=0; i<256; i++)
        if (plut==0)
            outport (DT2867_BASE+2062, i);
        else

```



```

        outport (DT2867_BASE+2062, 255-i);
    }
    outport (DT2867_BASE+1024, 0);
    Delay (1);
    for (i=0; i<256; i++) {
        outport (DT2867_BASE+1026, i);
        outport (DT2867_BASE+1026, i);
        outport (DT2867_BASE+1026, i);
    }
    outport (DT2867_BASE+1028, 0xFF);
    Delay (1);
    outport (DT2867_BASE+16, 0x00);
    Delay (1);
    outport (DT2867_BASE+18, 0x00);
    Delay (1);
    outport (DT2867_BASE+2064, 0x00);
    Delay (1);
    outport (DT2867_BASE+2066, 0x00);
    Delay (1);
    Live_image(0,0);
}
void TMyWindow::Live_image (int camera, int ilut)
{
    if (camera<0 || camera>3) camera=0;
    if (ilut<0 || ilut>7) ilut=0;
    outport (DT2867_BASE+4, DEF_4|ACQENB0);
    outport (DT2867_BASE+4, DEF_4|DISBUF1);
    outport (DT2867_BASE+1044, DEF_1044|(ilut<<8)|camera);
    outport (DT2867_BASE, DEF_0|DISPLAY);
}
void TMyWindow::Freeze_image (int camera, int ilut, int plut, int buffer)
{
    if (camera<0 || camera>3) camera=0;
    if (ilut<0 || ilut>7) ilut=0;
    if (plut<0 || plut>7) plut=0;
    if (buffer<0 || buffer>2) buffer=0;
    outport (DT2867_BASE, DEF_0|DISPLAY);
    outport (DT2867_BASE+1044, DEF_1044|(ilut<<8)|camera);
    if (buffer==0) {
        outport (DT2867_BASE+4, DEF_4|ACQENB0);
    }
    else if (buffer==1) {
        outport (DT2867_BASE+4, DEF_4|DISBUF0|ACQENB1);
    }
    else {
        outport (DT2867_BASE+2054, DEF_2054);
        outport (DT2867_BASE+2056, DEF_2056);
        outport (DT2867_BASE+2058, DEF_2058|(plut<<12));
        outport (DT2867_BASE+2048, 0x5001);
        outport (DT2867_BASE+2052, 0x5001);
        outport (DT2867_BASE+4, DEF_4|DISBUF1|DISBUF0|ACQENB2);
    }
}

```

```

        output(DT2867_BASE, DEF_0|DISPLAY|BUSY|MODE2);
        while ( inport(DT2867_BASE)&BUSY );
        if (buffer==2) {
            output(DT2867_BASE+2050, 0x3001);
            output(DT2867_BASE+2052, 0x3001);
        }
    }
void TMyWindow::Transfer_image(void)
{
    int x,y,i,j;
    unsigned Px;
    unsigned char pix;
    long sum;
    if(ib == NULL) return;
    for(y=0;y< num_rows;y++)
    {
        hist[y]=0;
    }
    output(DT2867_BASE,DEF_0);
    output(DT2867_BASE+4,DEF_4);
    while ( inport(DT2867_BASE)&BUSY );
    for(y=0;y< num_rows;y++)
    {
        output(DT2867_BASE+10,(y+100) & 0x1FF);
        output(DT2867_BASE+8,275);
        while ( inport(DT2867_BASE)&BUSY );
        for(x=0;x < num_rows;)
        {
            while ( inport(DT2867_BASE)&BUSY );
            Px = inport(DT2867_BASE+14);
            pix= *(ib[y]+x)= (Px & 0xff00)>>8;
            hist[pix]++;
            x++;
            pix= *(ib[y]+x)= Px & 0xff;
            hist[pix]++;
            x++;
        }
    }
    for(i=0;i<248;i++)
    {
        sum=0;
        for(j=i;j<(i+8);j++)
            sum+=hist[j];
        sum >>= 3;
        hist[i]= sum;
    }
    IFlag = TRUE;
    Invalidate(TRUE);
}

```

***runs all of the above defined functions in order to capture, crop, and histogram a pixel image**

```
void TMyWindow::Grab(void)
```

```

{
    Freeze_image(0,0,0,0);
    Transfer_image();
    Live_image(0,0);
}

```

***these functions are only called by the user when it is desired to display an image captured by an outside image processing software program: Global Lab. It is necessary to convert the tiff graphics file to an img graphics file in order to be used by the Borland C++ environment. Again the image is cropped around the centered bright first Purkinje reflection before it is displayed in the open graphics window.**

```

void TMyWindow::Convert(void)
{
    int y;
    struct tg
    {
        unsigned tagfun;
        unsigned tagtype;
        unsigned taglen;
        unsigned xxx;
        unsigned long val;
    } tag;
    int argc;
    char *argv[1], *env[1];
    FILE *outf;
    unsigned char pbuf[512];
    char instr[80];
    unsigned tifttype,tifver,ntags,i,si;
    unsigned long offset,pstrt,nxtln;
    unsigned ht,wd,gray={0},sz={256};
    int xoff,yoff,a,b,xbright=0,ybright=0;
    long ybias;
    unsigned char *buffer = new unsigned char[wd];
    char fname[30],drive[3],dir[66],name[9],ext[5];
    TFileOpenDialog::TData FilenameData(OFN_FILEMUSTEXIST | OFN_HIDEREADONLY
                                         | OFN_PATHMUSTEXIST, "Image Files
(*.tif) |*.tif|",0,
                                         "c:\\gl", "tif");
    if(TFileOpenDialog(this,FilenameData).Execute() != IDOK)
    {
        MessageBox("Cannot Open File","WARNING",MB_OK | MB_ICONSTOP);
        EFlag = TRUE;
    }
    else{
        (fp=fopen(FilenameData.FileName,"rb"));
        fnsplit(FilenameData.FileName,drive,dir,name,ext);
        strcpy(ext, ".img");
        strcpy(dir, "\\trident\\window");
        fnmerge(fname,drive,dir,name,ext);
        (outf=fopen(fname,"wb"));
    }
    if(fp == NULL)

```

```

{
    MessageBox("Cannot Open File","WARNING",MB_OK | MB_ICONSTOP);
    EFlag = TRUE;
}
fread(&tiftype,2,1,fp);
fread(&tifver,2,1,fp);
fread(&offset,4,1,fp);
if((tiftype != 0x4949) || (tifver != 0x2a))
{
    printf("\nWrong TIF file version");
    exit(0);
}
fseek(fp,offset,SEEK_SET);
fread(&ntags,2,1,fp);
for(i=0;i<ntags;i++)
{
    fread(&tag,sizeof(struct tg),1,fp);
    switch(tag.tagfun)
    {
        case 256:
            wd=tag.val & 0xffff;
            break;

        case 257:
            ht=tag.val & 0xffff;
            break;

        case 258:
            gray=tag.val & 0xffff;
            break;

        case 273:
            pstrt=tag.val;
            break;
    }
}
if(gray != 8)
{
    printf("\nNot a 256 gray level image");
    exit(0);
}
printf("\ninput picture width %d, height %d",wd,ht);
fseek(fp,pstrt,SEEK_SET);
for(i=0;i<ht;i++)
{
    fread(buffer,wd,1,fp);
    for(si = 0; si < wd; si++) {
        if((buffer[si]==255)&&(buffer[si+4]==255)&&(buffer[si+8]==255)&&
            (buffer[si+15]!=255)){
            xbright++;ybright=si+4;}
        if((buffer[ybright]==255)&&(xbright>=6)){
            xoff = si-128;
            yoff = i-128;
            xbright=si+4;
            ybright=i+4;
        }
    }
}

```

```

        i = ht;
        break;
    }
}
}
if((xoff+sz) > wd)
{
    xoff = wd-sz;
}
if((yoff+sz) > ht)
{
    yoff = ht-sz;
}
fseek(fp,pstrt,SEEK_SET);
ybias = (long)yoff * (long)wd;
nxtln= (long)(wd-sz);
fseek(fp,xoff+ybias,SEEK_CUR);
for(i=0;i<sz;i++)
{
    fread(&pbuf,sz,1,fp);
    fseek(fp,nxtln,SEEK_CUR);
    fwrite(&pbuf,sz,1,outf);
}
fclose(fp);
fclose(outf);
if(buffer) delete(buffer);
}
TMyWindow::~TMyWindow(void)
{
int y;
for(y=0;y<num_rows;y++)
    if(ib[y] != NULL)
    {
        delete(ib[y]);
        ib[y] = NULL;
    }
delete(ib);
ib = NULL;
for(y=0;y<num_rows;y++)
    if(ia[y] != NULL)
    {
        delete(ia[y]);
        ia[y] = NULL;
    }
delete(ia);
ia = NULL;
}

```

***this function sets up an open graphics window within which the 256x256 pixel image, the histogram, and the binarized 256x256 pixel image are displayed.**

```

void TMyWindow::Paint(TDC& , BOOL, TRect&)
{

```

```

TRect tr,hs;
int a,b=0,c=0,i,s,sum,x,y,xx,top=0,bottom=0;
int midx,midy,diff,fx,fy,yy,max=0,col=0,hh=0,dir=4;
int first[256]={0};
int second[256]={0};
float lg=0,dv;
unsigned char pix;
TClientDC tdc(*this);
TBrush Bbrush(tdc.GetBkColor());
    GetClientRect(tr);
    tr.left = 72;
    tr.right=328;
    tr.top = 19;
    tr.bottom=275;
    GetClientRect(hs);
    hs.left = 472;
    hs.right=728;
    hs.top = 19;
    hs.bottom=275;
    tdc.FillRect(tr,Bbrush);
    tdc.SelectObject(*Tpal);
    tdc.RealizePalette();
    if(IFlag)
    {
        for(y=0;y<num_rows;y++)
        {
            for(x=0;x<num_rows;x++)
            {
                if(ib[y] != NULL)
                {
                    pix = *(ib[y]+x);
                    pix >>= 1;
                    tdc.SetPixel(x+72,y+19,
                        PALETTEINDEX(pix+64));
                }
            }
        }
        tdc.FillRect(hs,Bbrush);
        if(IFlag)
        {
            for(y=0;y<num_rows;y++)
            {
                if(hist[y]>lg)
                {
                    lg = (float)hist[y];
                }
            }
            dv=lg/256.0;
            if(dv < 1) dv=1;
        }
    }

```

***the function for automatically determining the first peak in the histogram is developed here. The
 *range is set at 40 grey levels, ± 20 grey levels from the grey level of the peak. The 256x256 pixel
 *image is then binarized and the resulting image is displayed in the open graphics window**

```

for(i=4;(i<252)&&(hist[i]<=hist[i+6])&&(hist[i]>=hist[i-6])||(hist[i]<=lg/2.0);i++);
peak = i;
for(y=0;y<num_rows;y++)
{
    if(hist[y] !=NULL)
    {
        tdc.SelectObject(TPen(PALETTEINDEX(y/2+64)));
        tdc.MoveTo(y+472,275);
        tdc.LineTo(y+472,275-floor((float)hist[y]/dv));
    }
}
tdc.SelectObject(TPen(TColor(0,0,0)));
tdc.MoveTo(472,275);
tdc.LineTo(472,19);
tdc.MoveTo(472,275);
tdc.LineTo(728,275);
for(y=0;y<num_rows;y+=10)
{
    tdc.MoveTo(y+472,275);
    tdc.LineTo(y+472,280);
}
tdc.SelectObject(TPen(TColor(255,0,0),1,PS_DASHDOT));
tdc.MoveTo(472+peak-20,275);
tdc.LineTo(472+peak-20,19);
tdc.MoveTo(472+peak+20,275);
tdc.LineTo(472+peak+20,19);
tdc.RestorePalette();
for(y=0;y<num_rows;y++)
{
    for(x=0;x<num_rows;x++)
    {
        if(ib[y] != NULL)
        {
            pix = *(ib[y]+x);
            if((pix < (peak+20)) && (pix > (peak-20)))
            {
                *(ia[y]+x) = 1;
                tdc.SetPixel(x+72,y+300,TColor(0,0,0));
            }
            else
            {
                *(ia[y]+x) = 0;
                tdc.SetPixel(x+72,y+300,TColor(255,255,255));
            }
        }
    }
}
}

```

***now that the image has been binarized according to the range determined centered around the first peak in the histogram the next section of code runs the binarized image through a majority filter in order to smooth out the image to make it subsequently easier to run the contouring algorithm to define the pupil area**

```

for (s=0;s<5;s++)
{
    for(y=1;y<num_rows-1;y++)
    {
        for(x=1;x<num_rows-1;x++)
        {
            sum=(ia[y-1]+x);
            sum=sum+(ia[y]+x);
            sum=sum+(ia[y+1]+x);
            sum=sum+(ia[y-1]+x-1);
            sum=sum+(ia[y]+x-1);
            sum=sum+(ia[y+1]+x-1);
            sum=sum+(ia[y-1]+x+1);
            sum=sum+(ia[y]+x+1);
            sum=sum+(ia[y+1]+x+1);
            if(sum>4)
            {
                tdc.SetPixel(x+72,y+300,TColor(0,0,0));
                *(ia[y]+x) = 1;
            }
            else
            {
                tdc.SetPixel(x+72,y+300,TColor(255,255,255));
                *(ia[y]+x) = 0;
            }
        }
    }
}

```

***the next section of code contains the contouring algorithm used in defining the pupil area by entering the coordinates of the contour into two arrays.**

```

x=128;y=128;
while((*ia[y]+x)==0)&&(x<255)&&(y<255))
{
    x=x++;
    if(x==192){y++;x=128;}
    if(y==192)
    { x=128;y=128;
        while((*ia[y]+x==0)&&(x>3)&&(y>3))
        {
            x--;
            if(x==64){y--;x=128;}
            if(y==64)x=3;
        }
        x=255;
    }
}
while((*ia[y]+x)==1){x++;}

```



```

if(*(ia[y]+x)==0){b=x-1;c=y;}
x=b+10;y=y+10;
first[c]=b;
while(((x!=b)||y!=c))&&((x<255)&&(y<255)))
{
    if(hh==0){x=x-10;y=y-10;hh=10;}
    xx=x;yy=y;
    if(dir==1){x=x-1;y=y-1;}
    if(dir==2)y=y-1;
    if(dir==3){x=x+1;y=y-1;}
    if(dir==4)x=x+1;
    if(dir==5){x=x+1;y=y+1;}
    if((dir==6)||(dir==2))y=y+1;
    if((dir==7)||(dir==1)){x=x-1;y=y+1;}
    if((dir==8)||(dir==0))x=x-1;
    if(y<0)y=0;
    a=*(ia[y]+x);
    if(a==0){dir=dir+1;x=xx;y=yy;}
    if((a==1)&&(dir>0))dir=dir-8;
    if((a==1)&&(dir<1))dir=dir+5;
    if((a==0)&&(dir==9))dir=1;
    if((a==1)&&(first[y]==0))
    {
        first[y]=x;
        tdc.SetPixel(x+72,y+300,TCColor(255,255,0));
    }
    if((a==1)&&(first[y]!=0))
    {
        second[y]=x;
        tdc.SetPixel(x+72,y+300,TCColor(255,255,0));
    }
    if(x==255)x=0;
    if(y==255)y=0;
    if((x==b)&&(y==c)&&(dir!=4)){x=x+10;y=y+10;hh=0;}
}
for(y=0;y<num_rows;y++)
{
    if(first[y]>second[y])
    {
        s=first[y];first[y]=second[y];second[y]=s;
    }
}

```

***Now that the coordinate location of the pupil area has been defined the original grey levels of the**
***pupil area is searched for the brightest reflection below a grey level of 128 (half-intensity, necessary**
***to exclude locating the first Purkinje reflection unintentionally). This is the algorithm for locating**
***the fourth reflection**

```

max=0;s=0;
for(y=0;y<num_rows;y++)
{
    if(first[y]!=0)
    {

```

```

        for(x=first[y];x<second[y];x++)
        {
            while(*(ia[y]+x)==0){x++;}
            pix=tdc.GetPixel(x+72,y+19);
            if(pix<128)s=pix;
            else s=0;
            if(s>max){max=s;fx=x;fy=y;}
        }
    }
}
tdc.SetPixel(fx+72,fy+19,TCColor(255,255,0));
tdc.SetPixel(fx+72,fy+300,TCColor(255,255,0));
tdc.SetPixel(fx+72-1,fy+19,TCColor(255,255,0));
tdc.SetPixel(fx+72-1,fy+300,TCColor(255,255,0));
tdc.SetPixel(fx+72+1,fy+19,TCColor(255,255,0));
tdc.SetPixel(fx+72+1,fy+300,TCColor(255,255,0));
tdc.SetPixel(fx+72,fy+19-1,TCColor(255,255,0));
tdc.SetPixel(fx+72,fy+300-1,TCColor(255,255,0));
tdc.SetPixel(fx+72,fy+19+1,TCColor(255,255,0));
tdc.SetPixel(fx+72,fy+300+1,TCColor(255,255,0));
}
}

class TMyApp : public TApplication
{
    public:
        TMyApp() : TApplication() {}
        void InitMainWindow()
        {
            SetMainWindow(new TFrameWindow(0,"Image Window 1.0",
            new TMyWindow));
            GetMainWindow()->AssignMenu(1);
            nCmdShow = SW_SHOWMAXIMIZED;
        }
};

int OwlMain(int /*argc*/, char* /*argv*/ [])
{
    return TMyApp().Run();
}

```